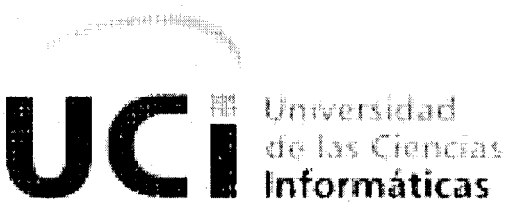
Universidad de las Ciencias Informáticas

Facultad 1



Título: Marco de trabajo .NET para la edición gráfica en páginas Web.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Yanirys Gallego Vera

Liber Francisco Matos Martín

Tutores:

Lic. Jeffrey Álvarez Massón

Lic. Denis Estévez Quesada

Ciudad de La Habana, julio, 2008

"Oye...
Escucha lo que las otras personas tienen que decir, es importante.
Sube...
Haz de los obstáculos escalones para aquello que quieres alcanzar.
Mas no te olvides de aquellos que no consiguieron subir en la escalera de la vida."

Charles Chaplin

Agradecimientos

Liber:

A mi tutor, el Lic. Jeffrey Álvarez Massón, por ser guía en la confección de este trabajo. A todas aquellas personas que de una forma u otra han contribuido con la realización de este trabajo y a mi formación profesional. A todos, gracias.

Yanirys:

A mi familia toda, por la confianza.

A mis tutores, por la certera guía, la dedicación y el apoyo brindado.

A mi grupo, por los momentos compartidos.

A mis profesores.

Dedicatoria

A mi madre,

Que tanto amor y apoyo me transmite.

A mi papá,

Que más que un padre ha sido un guía y un ejemplo para mí en todo momento.

A mi hermana, por enseñarme a afrontar todos los problemas con mucho valor y optimismo.

A Zhaskia, por estar siempre a mi lado, gracias por existir.

A toda mi familia siempre unida.

A mi amigo Chacón por su gran apoyo y ser mi primer maestro de informática.

Sin ustedes nada hubiese sido posible.

Liber.

A mi mamá, por todo el amor entregado, por nuestras lágrimas y sonrisas,

porque este también es su sueño.

A mi papá, por los momentos hermosos que llevo en el recuerdo.

A mi hermana y a Rosi, las dos flores de mi vida.

A Maya, por el ejemplo.

Yanirys.

RESUMEN

La edición y representación gráfica de datos constituyen dos de los retos principales que enfrentan los desarrolladores de aplicaciones Web, gran parte del tiempo necesario para la ejecución de proyectos es invertido en la implementación de funcionalidades o componentes capaces de brindar estas características así como en la preparación de personal con alto grado de conocimiento para hacer frente a estas tareas. Las librerías de componentes (controles) han sido una de las maneras utilizadas en los entornos de trabajo para solucionar esos problemas, no obstante dado el vertiginoso desarrollo de las tecnologías, las metodologías y los paradigmas en esta área, no siempre satisfacen los requerimientos actuales.

Desarrollar un componente que permita aunar un grupo de funcionalidades de acuerdo a las últimas tendencias en la Web tales como el intercambio de información de manera asincrónica y la personalización del contenido y su presentación por parte de los usuarios sin necesidad de tener grandes conocimientos de un lenguaje de programación o diseño de páginas Web, separando el diseño gráfico, las funcionalidades y la fuente de datos a representar, es el objetivo de este trabajo.

El diseño del control se basa en las nuevas características brindadas por el Framework de trabajo .NET 2.0 como el uso de los controles de orígenes de datos, el desarrollo de componentes basados en plantillas y controles personalizados que encapsulan diversas funcionalidades. Además se utilizan otras clases que complementan el desarrollo del control como son las clases diseñadoras del componente para gestionar su interfaz en tiempo de diseño y las convertidoras de tipo que facilitan la visualización de las propiedades.

PALABRAS CLAVE

- Framework
- Plantilla
- Enlace a datos
- Asincrónico
- Ajax
- Controles Web personalizados

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

INTRODUCCIÓN

La representación y edición gráfica de la información parece ser un fenómeno surgido con el desarrollo de las tecnologías, no obstante estudios demuestran que esta forma de representar el conocimiento ha acompañado al hombre desde hace miles de años evolucionando junto con él de acuerdo a sus necesidades, los medios técnicos, el desarrollo social, económico e informacional alcanzado en cada época.

Con el surgimiento de la computación el avance alcanzado en este campo se ha expandido hacia nuevas fronteras y retos. En los últimos años los volúmenes de información almacenada se han multiplicado de manera exponencial debido al desarrollo continuo y acelerado de los medios de almacenamiento, la interconexión entre los distintos dispositivos computacionales y la mayor posibilidad de acceso de las personas para consultar, publicar y compartir conocimientos en Internet.

El Ministerio del Interior (MININT) no está ajeno a esta problemática, el progresivo desarrollo alcanzado por sus medios tecnológicos ha hecho crecer de manera gradual la cantidad de información almacenada en soporte digital, traduciéndose esto en una mayor capacidad operativa y de respuesta a los servicios requeridos, lo constituye un nuevo reto para la institución. La necesidad de personal calificado, la dilatación en el desarrollo del software, el escaso nivel de interacción logrado entre los usuarios finales y los sistemas en desarrollo debido a que la representación de los datos se realiza mediante tablas y la baja reutilización de las funcionalidades, son algunas de las limitaciones que existen, entre otras cosas, como causas de las aún insuficientes herramientas de trabajo con que cuenta el ministerio.

Teniendo en cuenta lo planteado anteriormente se tiene para este trabajo el siguiente **problema de investigación**: La ausencia en el MININT de un marco de trabajo en .NET para la representación y edición gráfica de datos en las páginas web.

Con vista a resolver en cierta medida esta problemática se define como:

- **Objeto de estudio:** Herramientas para la representación y edición gráfica de datos en las páginas Web.
- **Campo de acción:** Herramientas para la representación y edición gráfica de datos en las páginas Web de aplicaciones ASP.NET desarrolladas por el MININT.

- **Objetivo general:** Implementar un marco de trabajo .NET que facilite la creación y utilización de herramientas para la presentación y edición de los datos de manera más intuitiva.

A partir de un análisis del objetivo general se derivan los siguientes **Objetivos específicos:**

- Investigar los marcos de trabajo existentes en .NET para la representación y edición gráfica de datos.
- Determinar funcionalidades generales no presentes en los marcos de trabajo estudiados para concebir herramientas que faciliten la representación y edición gráfica de datos.
- Diseñar un marco de trabajo .NET con componentes que mejoren la representación y edición gráfica en la Web.
- Definir extensiones que provean un comportamiento predeterminado para las componentes creadas y propicien la creación de nuevas extensiones.

Como **idea a defender:**

La implementación de un marco de trabajo .NET para la edición gráfica en páginas web facilitará la presentación e interacción con los datos de manera más intuitiva por parte de los usuarios.

El presente documento se estructura en cuatro capítulos y varios anexos, que incluye todo lo relacionado con el trabajo investigativo realizado, así como el diseño e implementación de la herramienta que se propone.

CAPÍTULO 1: APLICACIONES WEB Y METODOLOGÍA XP

1.1 Introducción

El surgimiento y desarrollo de las páginas Web, sus perspectivas futuras, el estado del arte sobre los marcos de trabajos existentes para la representación de datos en ellas y las nuevas metodologías de desarrollo del software son algunos de los temas a abordar durante el transcurso de este capítulo.

1.2 ¿Qué son las páginas Web?

Una página Web es un archivo de texto que contiene lenguajes de marcas de hipertexto (HTML), etiquetas de formato y vínculos a archivos gráficos, de video y de sonido, el cual se almacena en un servidor Web al que pueden acceder las computadoras que estén conectadas a él vía Internet o a través de una red LAN (1). Al archivo se puede acceder utilizando navegadores Web.

Hay dos propiedades de las páginas Web que las hacen únicas: son interactivas y pueden usar objetos multimedia. El término multimedia se utiliza para describir archivos de texto, sonido, animación y video que se combinan para presentar información. Cuando esos tipos de archivos se distribuyen por Internet o por una LAN, se puede utilizar el término hipermedia para describirlos. Gracias a la Web ya es posible disponer de multimedia a través de Internet.

1.3 Páginas estáticas y páginas dinámicas

Las páginas Web estáticas son aquellos sitios enfocados principalmente a mostrar una información permanente, donde el navegante se limita a obtener dicha información, sin que pueda interactuar con la página Web visitada, las Web estáticas están construidas principalmente con hipervínculos o enlaces (links) entre las páginas Web que conforman el sitio, y son incapaces de soportar funcionalidades dinámicas como mostrar resultados de consultas a bases de datos. Ofrecen pocas ventajas tanto a los programadores como a los usuarios, pues solo presentan texto plano acompañado de imágenes, y contenidos multimedia.

Las páginas Web dinámicas son aquellas que permiten crear aplicaciones dentro de la propia Web, otorgando una mayor interactividad con el navegante y una atención personalizada al cliente. Para su programación se necesitan otros lenguajes de programación que generen en su fase final el código HTML.

De acuerdo con el lugar donde se lleva a cabo el procedimiento de la página, es decir, la computadora que cargará con el peso adicional que supone que la página realice efectos y funcionalidades, las páginas dinámicas se clasifican en:

- Páginas dinámicas del cliente.
- Páginas dinámicas del servidor.

Enfocándose en cada una de ellas se encuentran la mayoría de los marcos de trabajo, componentes o librerías existentes. A continuación un análisis crítico de los más importantes para la edición y representación gráfica de datos en ambos.

1.3.1 Páginas dinámicas de cliente

Son las páginas dinámicas que se procesan en la computadora del cliente, cuya carga de procesamiento de los efectos y funcionalidades la soporta el navegador.

Usos típicos de las páginas de cliente son los efectos para la Web como rollovers, presentaciones en las que se pueden mover objetos por la página, validaciones de formularios, cálculos, etc. El código necesario para crear los efectos y funcionalidades se incluyen dentro del mismo archivo HTML en lenguajes script. Cuando una página contiene scripts de cliente, el navegador se encarga de interpretarlos y ejecutarlos para realizar los efectos y funcionalidades. Las páginas dinámicas de cliente se pueden escribir en lenguajes de programación como: JavaScript y VisuaBasicScript (VBScript).

Los lenguajes script tienen como ventaja el mejoramiento de la interacción con el usuario, además al ejecutarse en el cliente permiten un mayor rendimiento de los servidores al no sobrecargarlos de trabajo y dan la posibilidad de un tratamiento dinámico del contenido representado con menor tiempo de transición, pues no se tiene que reenviar el contenido de la página. Como desventaja, que se necesita de un conocimiento previo para su utilización porque carecen de un entorno gráfico y la no estandarización de la implementación del modelo DOM (Document Object Model) entre los diferentes navegadores utilizados dificulta su uso. Ejemplos:

Prototype: Librería JavaScript que incluye funcionalidades para el trabajo asincrónico con el servidor, también permite el manejo y optimización de arreglos, cadenas, objetos etc. Incluye métodos taquigráficos para hacer el código más legible, extiende el DOM, soporta JSON, y muchas otras características.

Rico: Librería de efectos AJAX de fácil uso para el intercambio de información asincrónica con el servidor. Dispone de funcionalidades para generar efectos gráficos como tablas actualizables y secciones de drag & drop. Es código abierto y está basada en prototype.js

Dojo Toolkit: Se basa en los mismos principios que Prototype, sin embargo, Dojo va más allá e introduce todo un conjunto de características como gráficos de vectores, un sistema de eventos muy completo, la capacidad de crear widgets (componentes basados en HTML + JavaScript) y las funciones para el intercambio de datos asincrónico. La desventaja de Dojo es que se ha vuelto sumamente complejo debido a su gran cantidad de funciones y su escasa documentación.

Dentro de los conceptos necesarios para comprender la creación de páginas dinámicas en el cliente están:

Cascading Style Sheets (CSS): "Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos". (2) Las hojas de estilo en cascada surgen por la limitación del lenguaje HTML a la hora de darle forma a un documento, permitiendo dar solución a estos problemas ya que son mecanismos simples que posibilitan aplicar formato a los documentos escritos en HTML como en lenguajes estructurados ejemplo XML, dando la posibilidad de separar el contenido de la presentación. Funciona mediante de reglas, o sea declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por varias de esas reglas aplicadas a un documento HTML o XML. Estas reglas constan de dos partes: un selector y la declaración.

Extensible Markup Lenguaje (XML): XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. (3) Tiene un papel fundamental en la actualidad ya que tiende a la globalización y a la compatibilidad entre los sistemas, siendo la tecnología que permite compartir la información de forma segura. Es un metalenguaje que permite definir lenguajes de presentación diferenciándolo del HTML que se centra en la representación de la información. Permite crear etiquetas según el contenido del documento siendo mucho más fácil de entender la representación. Un documento XML está integrado por elementos XML, cada uno tiene una etiqueta de inicio, una etiqueta de fin y los datos

comprendidos entre las etiquetas. "Es una tecnología muy sencilla que se vincula con otras tecnologías que la complementan y hacen que sea mucho más grande y con posibilidades mayores." (4)

JavaScript Object Notation (JSON): "JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos." (5)

Document Object Model (DOM):"El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos validos HTML y XML bien construidos. Define la estructura lógica de los documentos y el modo en que se accede y manipula." (6)

1.3.2 Páginas dinámicas de servidor

Las páginas dinámicas del servidor son reconocidas, interpretadas y ejecutadas por el propio servidor. Con ellas se puede hacer todo tipo de aplicaciones Web. Son muy útiles en trabajos donde se tiene que acceder a la información centralizada, situada en una base de datos en el servidor y cuando por razones de seguridad los cálculos no se pueden realizar en la computadora del usuario.

Es importante destacar que las páginas dinámicas de servidor son necesarias porque para crear la mayoría de las aplicaciones Web se debe tener acceso a muchos recursos externos a la computadora cliente, principalmente en bases de datos alojadas en otros servidores. Las páginas dinámicas de servidor se pueden escribir en lenguajes de programación como, Common Gateway Interface (CGI) comúnmente escritos en Perl, Active Server Pages (ASP), Hipertext Procesor (PHP), Java Server Pages (JSP) y especialmente ASP.NET, que será el lenguaje utilizado para la realización del trabajo en este caso.

Una de las ventajas de este tipo de programación es que el cliente no puede ver el código que realiza las operaciones más importantes, ya que se ejecutan en el servidor y se transforman en HTML antes de ser enviado al navegador. Además permite el acceso a datos dinámicamente.

Como desventaja, se requiere de un servidor más potente y con más capacidades que el que se necesita para las páginas de cliente. Además, estos servidores podrán soportar menos usuarios concurrentes, porque se demandará más tiempo de procesamiento para la atención de cada uno. A continuación algunos de los componentes y librerías de ASP.NET estudiados como referencia para este trabajo.

- **Gridview, Formview, Listview:** Son controles enlazados a datos que definen su interfaz mediante plantillas. Tienen como ventaja que separan la lógica de acceso a datos, la interfaz y las funcionalidades del control. Su principal desventaja es que presentan los datos en tablas y no cuentan con funciones para el manejo de los mismos en el cliente.

- **Microsoft ASP.NET AJAX:** Gira entorno al lenguaje de programación Web ASP.NET. Ofrece una solución completamente integrada a .NET Framework 2.0 expuesta a través de controles de lado del servidor. Posee una librería de componentes visuales vinculados a la gestión de la transmisión de la información de manera asincrónica entre la página en el cliente y el servidor. Permite la creación de controles del servidor extendidos con funcionalidades JavaScript mediante la definición de nuevos controles personalizados.

1.4 Evolución tecnológica de Internet

El desarrollo tecnológico en Internet se puede sintetizar, a grandes rasgos, en tres etapas o momentos. Una primera etapa que podemos ubicar hasta el año de 1997, una segunda que correspondió a los años de 1998 al 2001 y la etapa actual o tercera que comenzó en el año 2002.

La primera etapa se basó en la creación de llamativas páginas Web estáticas, las cuales se trasladaban desde servidor hasta el cliente utilizando el protocolo http, sin que se realizara ningún proceso adicional sobre las páginas. En este momento se desarrollaron las herramientas de creación y visualización de páginas Web, las cuales exigían el conocimiento del lenguaje HTML y el protocolo de transferencia http.

La segunda etapa se caracterizó por la inclusión de varias tecnologías, tanto en el ámbito del cliente como en el servidor. En el lado del cliente se pusieron a disposición de la creación de páginas Web una gran variedad de controles ActiveX, Applet, lenguaje HTML dinámico, JScript y VBScript, entre otros elementos. En el lado del servidor se brindó la posibilidad de resolver ciertas partes de la página Web en forma dinámica, en el momento en que se efectuara una petición de la página,

mediante la creación de nuevos lenguajes y componentes que propiciaron un desarrollo sencillo. Dentro de las tecnologías empleadas en esta etapa se encuentran:

- Controles ActiveX: Son módulos de código que representan la implementación de un control que realiza una función determinada y que pueden ser utilizados dentro de las aplicaciones Windows y Web. Además se pueden transferir a través de la red y ejecutarse por el explorador Internet Explorer bajo el sistema operativo Windows.

- Applet: Son módulos similares a ActiveX. Se crean con el lenguaje Java, pero para su utilización se requiere contar con una máquina virtual de Java.

- El lenguaje HTML: Se emplea universalmente en Internet para definir en un único archivo los datos y la presentación de una página.

- ASP: Es una tecnología de Microsoft que ofrece el acceso, mediante programación, a los recursos del servidor. Además brinda la posibilidad de resolver secciones de una página antes de que se envíe al cliente.

- Hyper Text Transfer Protocol (http): Es un protocolo que se emplea a través de la red para la transferencia de información en forma de código HTML.

- VBScript, JavaScript y JScript: Son lenguajes de programación creados para formular páginas y aplicaciones Web, los que resultan ser un subconjunto de los respectivos lenguajes de programación originales.

- PHP y Perl: Son lenguajes de programación que se crearon de forma exclusiva para la programación de páginas Web dinámicas. Ambos lenguajes son muy parecidos al C y se pueden emplear en IIS mediante la inclusión de agregados que hayan sido suministrados por terceros fabricantes.

- JavaServer Pages (JSP): Es una tecnología abierta que ofrece acceso mediante programación a los recursos del servidor. Además permite secciones de una página antes de que esta sea enviada al cliente. La principal diferencia con ASP reside en que existen varios proveedores que le ofrecen soporte.

La tercera etapa se desarrolla a partir del año 2002. En ella están presentes con plena vigencia todas las tecnologías y herramientas que se utilizaron en la segunda etapa para la creación de páginas

y aplicaciones Web, pero se agregan nuevas tecnologías y herramientas como ASP.NET y XML, con el fin de propiciar una mayor integración de las aplicaciones a la red y a los dispositivos móviles.

1.5 Aparición de ASP.NET

Lo distintivo de la tercera etapa evolutiva de Internet fue la aparición de ASP.NET. Esta tecnología no surgió de golpe, sino que es producto del desarrollo lógico de las formas de programar para Internet.

No fue hasta la aparición de la versión 5 de Visual Basic, en que se incursionó en la programación para Internet, a través de los llamados documentos ActiveX. En realidad esta tecnología se ofreció como una opción rápida para integrar una aplicación estándar con Internet y como respuesta al auge de las tecnologías propuestas por Sun Microsystems.

Los documentos ActiveX permitían a las aplicaciones Windows que tomaran algún beneficio de las características del explorador de Internet. Básicamente este último servía como contenedor de la aplicación, pero las ventajas obtenidas no iban más allá de esa función y no se obtenía una integración real entre la Web y la aplicación. Ello provocó que esta tecnología casi no tuviera seguidores. Después Microsoft creó, como solución a las limitaciones anteriores, Visual InternetDev, el cual era un producto que permitía escribir páginas como secciones a ser ejecutadas por el servidor Internet Information Server (IIS) y además contenía varios controles a los efectos de facilitar la construcción de interfaces gráficas.

De inmediato los programadores se encontraron con que la mayoría de las aplicaciones Windows debían ser diseñadas si se deseaba migrar al modelo Web, además de que tenían que empezar a aprender a utilizar los nuevos objetos, hacer uso de sus principales propiedades y eventos y comprender con mayor profundidad el lenguaje de marcación HTML. Muchos optaron por emplear las herramientas propuestas por la compañía Sun, como el lenguaje Java. Sin embargo, los problemas seguían latentes, pues existían grandes limitaciones para realizar tareas sencillas, o lograr que una misma aplicación se ejecutara en diferentes plataformas de forma correcta.

Ante los problemas que se continuaron presentando para integrar las aplicaciones Web y los constantes reclamos de la comunidad de programadores, Microsoft incluyó en Visual Studio 6 dos nuevas características que estaban directamente relacionadas con la programación para Internet, ellas son: un diseñador para páginas DHTML y las aplicaciones IIS.

El Diseñador para páginas DHTML permitía explorar los recursos del cliente mediante la creación de aplicaciones que utilizan un grupo de objetos establecidos por Internet Explorer. Microsoft básicamente había dotado al explorador Internet Explorer de un conjunto de bibliotecas que era factible de ser empleado desde código script en la página HTML. A esta tecnología se le llamó HTML dinámico, ya que gran parte del conjunto de objetos brindaba la posibilidad de ejecutar tareas de forma dinámica, sin necesidad de realizar una nueva petición al servidor. Si bien esta característica fue de utilidad para los programadores, la realidad fue que el conjunto de objetos propuestos por Microsoft y Netscape, con diferencias importantes, hacía difícil la creación de aplicaciones compatibles para ambos exploradores. Por ello esta opción fue elegida por aquellos que necesitaban crear aplicaciones para entornos controlados, como los casos de las Intranet.

La propuesta de las aplicaciones IIS posibilita a los programadores crear una aplicación ejecutable, la cual debía ser instalada en el servidor de Internet y sería vista por el navegador como una página estándar, aunque en realidad se trataba de una página de resolución dinámica (ASP). Básicamente, cuando el servidor recibía una petición del explorador, se ejecutaba la biblioteca y su respuesta era enviada otra vez al cliente. Esto permitía emplear los recursos del servidor para realizar una gran cantidad de funciones.

A pesar de que las aplicaciones IIS constituyeron los primeros pasos en integración real con Internet, contaban con algunas restricciones significativas. Una de ellas era que las páginas a ser resueltas en forma dinámica debían estar incluida en el archivo ejecutable, por lo que hacía difícil su posterior modificación. Por otro lado, el modelo completo de objetos ofrecidos por el servidor no siempre estaba disponible para una aplicación IIS y la tarea de la construcción de la interfaz gráfica era realmente tediosa en comparación con otros candidatos como Visual InterDev. Fue Visual InterDev la herramienta más empleada por los programadores para Internet, en tanto ofrecía la posibilidad de implementar verdaderas soluciones Web, permitía escribir páginas con secciones que debían ser resueltas en forma dinámica por el servidor igual que las aplicaciones IIS, brindaba controles para facilitar la creación de la interfaz gráfica y enlazaba los controles a datos. No obstante, esta herramienta poseía tres desventajas:

- Las aplicaciones resultantes no eran compiladas, sino interpretadas, por lo que el resultado final tenía siempre un rendimiento inferior al de un ejecutable real.

- Las funcionalidades residían dentro de la página como elementos independientes, lo que dificultaba la reutilización del código, es decir el código generalmente estaba repetido en varias secciones de la página y mezclado con información de presentación.

- Las herramientas para la depuración del código escrito en la aplicación no siempre funcionaban correctamente, por lo que se necesitaba emplear tiempo adicional para hacer una revisión manual de todo el código.

Ante el panorama descrito, Microsoft decidió comenzar a trabajar en la solución de todos los puntos negativos de estas tecnologías y herramientas para la integración de Internet, a fin de lograr un modelo más conveniente y consistente, el cual incluiría además, lo mejor de todo lo aplicado con anterioridad. El resultado final apareció a comienzos del año 2002, y se nombró ASP.NET.

ASP.NET mejora diversos aspectos con respecto a su antecesor directo ASP, las cuales aceleran los tiempos de programación y ofrecen un modelo más consistente. Visto desde el punto de Visual Studio.NET, se elimina Visual InterDev como producto, aunque se incluyen todas sus características en el nuevo entorno de desarrollo.

1.6 ¿Qué es ASP.NET?

ASP.NET es una plataforma de programación Web unificada que proporciona los servicios necesarios para que los programadores creen aplicaciones Web para el sistema empresarial moderno. Si bien ASP.NET es en gran medida compatible con la sintaxis ASP, proporciona también un modelo de programación y una estructura nueva para crear aplicaciones más seguras, escalares y estables.

"Además, ASP.NET es un entorno compilado basado en la plataforma .NET por lo que se integra a tecnologías como Visual Basic.NET, C#, JScript.NET. Los programadores pueden aprovechar las ventajas de estas tecnologías, que incluyen el Common Language Runtime administrado, seguridad de tipos, herencia, etcétera." (1)

ASP.NET se ha diseñado para funcionar sin problemas con los editores HTML WYSIWYG y otras herramientas de programación como Visual Studio.NET, hace más fácil la programación Web y ofrece todas las ventajas de estas herramientas, con una interfaz gráfica de usuario que los programadores pueden usar para ubicar controles de servidor en una página Web e integrar completamente la compatibilidad con la depuración de errores.

Los programadores pueden elegir uno de los dos modelos que se emplean para crear una aplicación ASP.NET: formularios Web y servicios Web – o su combinación en dependencia de sus objetivos –. Ambos son compatibles con la misma infraestructura, que permite utilizar esquemas de autenticación, almacenar en caché datos que se usan con frecuencia y personalizar la configuración de la aplicación, entre otras ventajas.

Los formularios Web permiten crear páginas Web basadas en formularios muy eficaces, en los cuales se pueden usar controles de servidor ASP.NET para crear elementos comunes de la interfaz gráfica de usuario y programarlos para que realicen las tareas comunes. Estos controles permiten crear con rapidez un formulario Web a partir de componentes integrados reutilizables o personalizados, con un código de página simplificado.

Un servicio Web XML proporciona los medios para acceder a la funcionalidad del servidor de manera remota. Con los servicios Web, las empresas pueden exponer interfaces de programación a sus datos o lógica empresarial, que, a su vez, pueden obtener y manipular las aplicaciones de cliente y servidor. Los servicios Web XML permiten el intercambio de datos en escenarios cliente – servidor o servidor – servidor, utilizando estándares como los servicios de mensajería HTTP y el lenguaje XML para que los datos pasen los servidores de seguridad. Los servicios Web XML no están ligados a ninguna tecnología de componentes ni a ninguna convención de llamada a objetos concreta. En consecuencia, se puede acceder a ellos mediante los programas escritos en cualquier lenguaje, que empleen cualquier modelo de componentes y que se ejecuten en cualquier sistema operativo.

Cada uno de estos modelos puede aprovechar al máximo todas las características de ASP.NET y es un marco de trabajo de programación generado en el Common Language Runtime, que puede emplearse en un servidor para generar eficaces aplicaciones Web. ASP.NET ofrece varias ventajas importantes, entre las que se encuentran:

- *Mejor rendimiento.* ASP.NET es un código del Common Language Runtime compilado que se ejecuta en el servidor. A diferencia de sus predecesores, ASP.NET puede aprovechar las ventajas del enlace anticipado, la compilación *just-in-time*, la optimización nativa y los servicios de caché desde el primer momento.

- *Compatibilidad con herramientas de primer nivel.* El marco de trabajo de ASP.NET se complementa con un diseñador y una caja de herramientas muy completos en el Entorno Integrado de Programación (IDE) de Visual Studio.NET. la edición WYSIWYG, los controles

de servidor de arrastrar y colocar, y la implementación automática son solo algunas de las características que proporciona esta eficaz herramienta.

- *Eficacia y flexibilidad.* Debido a que ASP.NET se basa en el Common Language Runtime (CLR), la eficacia y la flexibilidad de toda esa plataforma se encuentra disponible para los programadores de aplicaciones Web. La biblioteca de clases de .NET Framework, la Mensajería y las soluciones de Acceso a datos con el empleo de ADO.NET se encuentran accesibles desde el Web de manera uniforme. ASP.NET es también independiente del lenguaje de programación, por lo que puede elegir el lenguaje que mejor se adapte a la aplicación o dividir la aplicación en varios lenguajes. Además, la interoperabilidad del Common Language Runtime garantiza que la inversión existente en programación basada en COM se conserve al migrar a ASP.NET.

- *Simplicidad.* ASP.NET facilita la realización de tareas comunes, desde el sencillo envío de formularios y la autenticación del cliente, hasta la implementación y la configuración de sitios. Por ejemplo, el marco de trabajo de página de ASP.NET permite generar interfaces de usuario, que separan claramente la lógica de aplicación del código de presentación y controlar eventos en un sencillo modelo de procesamiento de formularios de tipo Visual Basic.NET. Además, Common Language Runtime simplifica la programación con servicios de código administrado como el recuento de referencia automático y el recolector de elementos no utilizados.

- *Facilidad de uso.* ASP.NET emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la configuración al entorno de servidor y las aplicaciones Web. Debido a que la información se configuración se almacena como texto sin formato, se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local. Esta filosofía de "administración local cero" se extiende, asimismo, a la implementación de las aplicaciones ASP.NET Framework en un servidor, lo que se logra mediante la copia de los archivos que este necesita. No se requiere el reinicio del servidor, ni siquiera para implementar o reemplazar el código compilado en ejecución.

- *Escalabilidad y disponibilidad.* ASP.NET se ha diseñado teniendo en cuenta estas características, con el fin de mejorar el rendimiento en entornos agrupados y de múltiples procesadores. Además, el motor de tiempo de ejecución de ASP.NET controla y administra los procesos de cerca, por lo que si uno no se comporta de forma adecuada (filtraciones,

bloqueos), se puede crear un proceso nuevo en su lugar, lo que ayuda a mantener la aplicación disponible constantemente para controlar solicitudes.

- *Posibilidad de personalización y extensibilidad.* ASP.NET presenta una arquitectura bien diseñada que permite a los programadores insertar su código en el nivel adecuado. De hecho, es posible extender o reemplazar cualquier subcomponente del motor de tiempo de ejecución de ASP.NET con su propio componente personalizado. La implementación de la autenticación personalizada o de los servicios de estado nunca ha sido más fácil.

- *Seguridad.* Con la autenticación de Windows integrada y la configuración por aplicación, se puede tener la completa seguridad de que las aplicaciones están a salvo.

1.7 ¿Qué son las aplicaciones Web?

Visual Studio.NET permite crear aplicaciones que aprovechan la potencia de la Web, desde un sitio Web tradicional que sirva páginas HTML, hasta aplicaciones completas de negocios que proporcionen componentes basados en Web para el intercambio de datos mediante la tecnología XML y que se ejecuten en una Intranet o en Internet.

"En pocos años la Web ha evolucionado enormemente: se ha pasado de páginas sencillas, con pocas imágenes y contenidos estáticos a páginas complejas con contenidos dinámicos que provienen de bases de datos, lo que permite la creación de "aplicaciones web". De forma breve, una aplicación web se puede definir como una aplicación en la cual un usuario por medio de un navegador realiza peticiones a una aplicación remota accesible a través de Internet (o a través de una intranet) y que recibe una respuesta que se muestra en el propio navegador. (7)"

Una aplicación Web de Visual Studio.NET se genera alrededor de ASP.NET, que al formar parte de .NET Framework proporciona acceso a todas las funciones de este marco de trabajo. Por ejemplo:

- Puede crear aplicaciones Web ASP.NET mediante cualquier lenguaje de programación .NET, ya sea Visual Basic, C#, Extensiones administradas de C++, etcétera.

- Puede emplear las utilidades de depuración de excepciones incluidas en la plataforma.

- Para el acceso a los datos se utiliza ADO.NET, una tecnología incluida en la propia plataforma.

De forma similar, puede tener acceso a los servicios del sistema operativo y al resto de los elementos de la plataforma mediante clases .NET Framework.

Como con cualquier aplicación .NET, si dispone de .NET Framework puede crear aplicaciones ASP.NET mediante editores de texto, un compilador de línea de comandos y otras herramientas sencillas. Puede copiar archivos manualmente en Internet Information Server (IIS).

La ventaja del uso de Visual Studio reside en que proporciona herramientas que facilitan el desarrollo de aplicaciones de forma más rápida y fiable, entre las que se encuentran:

- Diseñadores visuales para páginas Web con controles para arrastrar y colocar, y vistas de código HTML con comprobación de sintaxis.
- Editores de códigos inteligentes que incluyen finalización de instrucciones, comprobación de sintaxis y otras características de *IntelliSense*.
- Compilación y depuración integradas.
- Utilidades de administración de proyectos para la creación y administración de archivos de aplicación, incluida la implementación en servidores locales o remotos.

Por lo general las aplicaciones que se crean para que funcionen en Internet son aplicaciones distribuidas, o sea se crean para que funcionen en el servidor y se pueden acceder a ellas desde un equipo cliente que se conecte al mismo.

1.8 Funcionamiento de las aplicaciones Web

Las aplicaciones Web ASP.NET se ejecutan en un servidor Web configurado con Microsoft Internet Information Server, aunque no se necesita trabajar directamente con él porque se pueden programar utilidades mediante clases ASP.NET. Cuando es necesario Visual Studio.NET controla las tareas de administración de archivos, tales como la creación de aplicaciones IIS y proporciona los medios para implementarlas.

En términos generales cuando se crea una aplicación Web esta siempre posee una interfaz gráfica a la cual accede el usuario a través de una URL o un hipervínculo situado en cualquier otra página. Una vez presente el formulario o página Web de acceso de la aplicación, esta hace la llamada al servidor para satisfacer la demanda de la información solicitada por el usuario empleando el

protocolo http y el lenguaje de intercambio de datos XML. En el servidor, usando los elementos programados y situados en el nivel de negocio, accede a las bases de datos y realiza el proceso de información correspondiente a la solicitud o emplea uno de los servicios Web XML que se pudiera haber creado para ejecutar la acción solicitada. Luego vuelve a crear una nueva página Web con los resultados exigidos, los cuales se visualizan de nuevo en el navegador.

Este funcionamiento maneja uno de los conceptos principales que encierra la forma de ejecución de las aplicaciones Web: la división del trabajo entre el cliente y el servidor, es decir del lado del cliente el navegador presenta el formulario al usuario y este interactúa con él, causando que el formulario se envíe de vuelta al servidor. Sin embargo, como todo procesamiento que interactúa con componentes de servidor, debe ocurrir en el servidor, o sea, para cada acción que necesita procesamiento el formulario debe enviarse al servidor, procesarse y ser devuelto al navegador (cliente).

En los formularios Web, la mayoría de las acciones del usuario, como hacer clic en un botón, provoca una acción de ida y vuelta, por lo que los eventos disponibles en los controles de servidor ASP.NET suelen limitarse a eventos de tipo Click, que requieren una respuesta explícita del usuario. Por esa razón, los controles de servidor no muestran los eventos que se producen con mucha frecuencia, como por ejemplo, `onmouseover`.

En cualquier escenario Web, las páginas se vuelven a crear con cada acción de ida y vuelta. Tan pronto como el servidor termina de procesar y enviar la página al navegador Web, descarta la información de la página. Mediante la liberación de recursos de servidor después de cada petición, se puede ajustar una aplicación Web para que admita cientos o miles de usuarios simultáneos. La siguiente vez que se envía la página, el servidor empieza a crearla y procesarla y, por esta razón, se dice que las páginas Web no tienen estado, ya que los valores de las variables y controles de una página no se conservan en el servidor.

1.9 Metodologías ágiles

Las metodologías ágiles constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales (ISO-9000, CMM, etc.) debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

"Las Metodologías Ágiles valoran, Al individuo y las interacciones en el equipo de desarrollo más que a las actividades y las herramientas. Desarrollar software que funciona más que conseguir una buena documentación, implica minimalismo respecto del modelado y la documentación del sistema. La colaboración con el cliente más que la negociación de un contrato. Responder a los cambios más que seguir estrictamente una planificación." (8)

1.9.1 ¿Por qué surgen las Metodologías Ágiles?

Dificultad para implantar metodologías tradicionales. Sofisticadas herramientas CASE y notaciones (UML). Una solución a medida para un segmento importante de proyectos de desarrollo de software. Aceptar el cambio.

1.9.2 Metodología XP (eXtreme Programing)

"XP es un método ligero, para equipos de desarrollo de software de tamaño pequeño a mediano, que se enfrentan a requisitos vagos y que cambian rápidamente". (9)

XP forma parte del conjunto de métodos ágiles .Surge como nueva disciplina de desarrollo de software aproximadamente en 1996. Está centrada en potenciar las relaciones interpersonales, el trabajo en equipo, el aprendizaje de los desarrolladores, propiciar un buen clima de trabajo, la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. XP ha venido a situarse en el centro las nuevas tendencias dentro de las metodologías para el desarrollo de software generando grandes pasiones y controversias dentro de la comunidad internacional.

Valores, prácticas y principios de la metodología XP

La programación extrema es una metodología de desarrollo ligera basada en una serie de valores, principios y una docena de prácticas que propician un aumento en la productividad a la hora de generar software.

1.9.2.1 Valores que promueve

Los valores que promueve la metodología XP son requisitos indispensables para el éxito de un proyecto de desarrollo de software. En primer lugar la comunicación. Algunos problemas en los proyectos tienen su origen en este aspecto. XP ayuda mediante sus prácticas a fomentar la comunicación.

La sencillez, XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación al proceso y la codificación. Es mejor hacer hoy algo simple, que hacerlo complicado y probablemente nunca usarlo. La sencillez y la comunicación se complementan, cuanto más simple es el sistema menos se tiene que comunicar de él.

Retroalimentación concreta y frecuente del cliente del equipo y de los usuarios finales, es el tercer valor, con ella se da una mayor oportunidad de dirigir el esfuerzo eficientemente por medio de pruebas funcionales al software.

La **valentía** de asumir retos, afrontar los problemas es una de las condiciones necesarias en la dinámica de desarrollo con XP.

1.9.2.2 Principios

Los cuatro valores dan origen a cinco principios básicos:

- Conseguir retroalimentación rápida.
- No complicar las cosas con suposiciones (asumir que las cosas son simples).
- Realizar cambios incrementales.
- Abrazar el cambio.
- Generar productos de calidad.

Los cinco principios se manifiestan a través de las prácticas de la programación extrema suponiendo un puente entre los valores y las prácticas.

1.9.2.3 Prácticas

Las prácticas utilizadas en esta metodología no son nuevas, muchas de ellas se han utilizado durante años y han sido catalogadas como buenas, solo que en XP son llevadas al extremo de manera que se obtenga algo mejor que la suma de sus partes.

El juego de la planificación

El alcance de la siguiente versión está definido por las consideraciones de negocios (prioridad de los módulos, fechas de entrega) y estimaciones técnicas (estimaciones de funciones, consecuencias).

El objetivo es maximizar el valor del software producido, la estrategia es poner en producción las características más importantes lo antes posible, las piezas clave son las historias de usuarios.

Responsabilidades:

Negocio:

- Definir el alcance para esta versión del software.
- Definir las historias que se harán primero.
- Fijar tiempos y fechas de entrega.

Técnico:

- Decir cuan larga puede ser cada historia.
- Estimar el tiempo necesario para cada historia de usuario.

Versiones pequeñas

Un sistema simple se pone rápidamente en producción. Periódicamente, se producen nuevas versiones agregando en cada iteración aquellas funciones consideradas valiosas para el cliente.

Metáfora del sistema

Cada Proyecto es guiado por una historia simple de cómo funciona el sistema en general, reemplaza a la arquitectura y debe estar en lenguaje común, entendible para todos, cliente y desarrolladores, esta puede cambiar permanentemente.

Diseño simple

El sistema se diseña con la máxima simplicidad posible Se plasma el diseño en tarjetas CRC (Clase – Responsabilidad - Colaboración), no se implementan características que no son necesarias.

Pruebas continuas

Los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.

Refactorización

Reestructurar el sistema sin cambiar su comportamiento eliminando código duplicado, simplificando funciones, mejorando el código constantemente, si el código se está volviendo complicado se debería modificar el diseño y volver a uno más simple.

Programación por parejas

El código es escrito por dos personas trabajando en la misma máquina.

Posesión colectiva del código

Cualquier programador puede cambiar cualquier parte del sistema en cualquier momento.

Integración continua

Los cambios se integran en el código base varias veces por día. Todos los casos de prueba se deben pasar antes y después de la integración, se dispone de una máquina para la integración y se realizan test funcionales en donde participa el cliente. La integración continua a menudo reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido. Se recomienda el uso de repositorios para facilitar esta tarea.

Semana laboral de 40 horas

Cada trabajador ejerce su tarea no más de 40 Horas por semana. Si fuera necesario hacer horas extra, esto no debería hacerse dos semanas consecutivas.

Cliente en el sitio

El equipo de desarrollo tiene acceso todo el tiempo al cliente, el cual está disponible para responder preguntas, fijar prioridades, etc. Esto no siempre se consigue, un cliente con poca experiencia no sirve y un cliente muy experimentado no es disponible.

Estándares de codificación

Todo el código debe estar escrito de acuerdo a un estándar de codificación que facilite la comunicación.

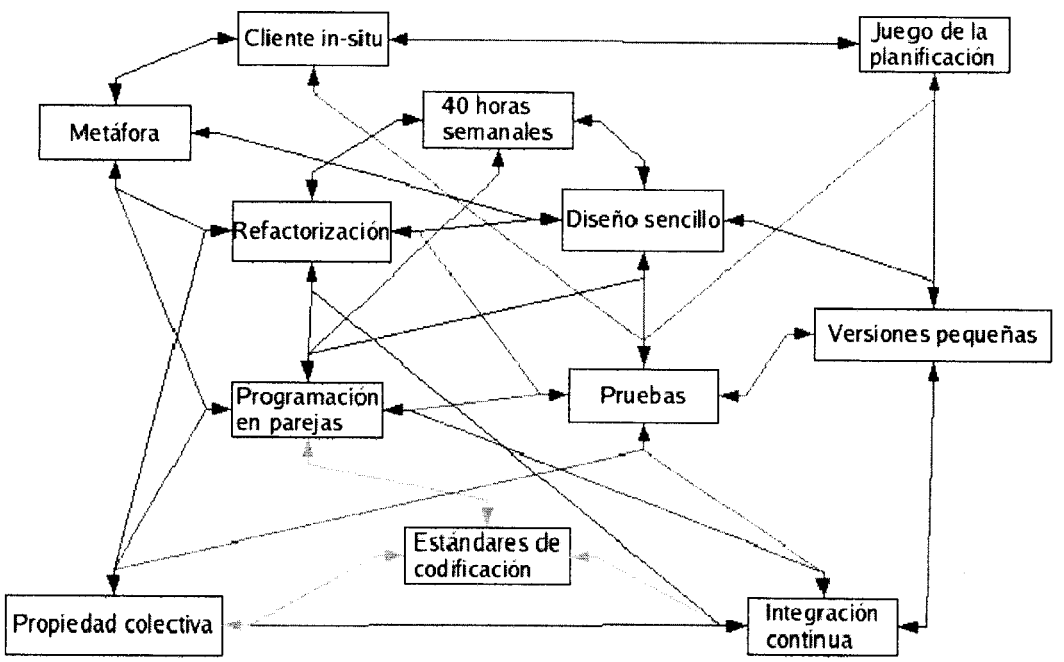


Figura 1.1 Diagrama de interación de las práctica XP.

Las cuatro variables

XP define cuatro variables para la realización de cualquier proyecto: coste, tiempo, calidad y alcance. De estas cuatro variables, sólo tres de ellas podrán ser fijadas por las fuerzas externas al proyecto (clientes y jefes de proyecto), mientras que el valor de la cuarta variable será establecida por el equipo de desarrollo en función de los valores de las otras tres.

1.9.2.4 Fases del desarrollo

Estas fases son el ciclo de vida para el desarrollo de un proyecto en XP.

Fase I. Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II. Planificación de la entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según el alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Fase III. Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar

establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración para maximizar el valor de negocio. Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del plan de la iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

Fase IV. Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación por ejemplo, durante la fase de mantenimiento.

Fase V. Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

1.10 Conclusiones

En este capítulo se abordaron conceptos referentes a las páginas Web, ¿cuáles son las páginas Web estáticas y las dinámicas?; ¿qué son las aplicaciones Web y cómo funcionan?, así como la evolución tecnológica de Internet, la aparición de la tecnología ASP.NET y lo que ella representa para la programación de aplicaciones para Internet. Se hace alusión al estado del arte relacionado con la existencia de componentes y librería tanto para el trabajo de lado del cliente como el servidor, las cuales facilitan el diseño de páginas Web. Además se hace referencia en las metodologías ágiles,

haciendo especial énfasis en la metodología Programación Extrema, detallando aspectos fundamentales que la caracterizan, así como sus fases y prácticas.

CAPÍTULO 2: APLICACIÓN DE LAS FASES EXPLORACIÓN Y PLANIFICACIÓN DE ENTREGAS

2.1 Introducción

En este capítulo se aplica la metodología XP para la confección de un control Web personalizado para el desarrollo de páginas Web, que tiene como objetivo fundamental la representación y edición de datos de manera gráfica. En específico las fases de exploración y planificación. En la primera se hace un estudio de la metáfora del sistema, las herramientas a utilizar para el desarrollo del software y los aspectos teóricos básicos. Durante la segunda, se definen las historias de usuarios, sus costos y los objetivos de cada iteración.

2.2 Fase de exploración

2.2.1 Metáfora del sistema

El programador sitúa el componente en el área de trabajo y este le brinda la posibilidad de definir la fuente de datos a mostrar, la fuente de datos para persistir el estado y de configurar su interfaz gráfica, dándole soporte tanto para el trabajo en el lado del servidor como para el lado del cliente.

2.2.2 Herramientas

Las herramientas a utilizar de acuerdo a los intereses del cliente y las prestaciones de servicios necesarios para la elaboración del programa son:

- **Visual Studio .NET 2005:** Es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Permite el desarrollo en varios lenguajes, Visual Basic, Visual C++, Visual C# y Visual J#, que utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML. Además incluye dentro de su entorno de desarrollo amplias posibilidades de configuración y gestión de controles en tiempo de diseño así como para la ampliación y creación de los mismos.

- **NoteZilla 7.0:** Es un software utilizado para gestionar tareas, con todo un completo conjunto de funciones para poder organizarlas al estilo de trabajo XP. Además de contar con un navegador para su administración y control total.

- **Subversion:** Es un sistema centralizado para compartir información. Permite guardar archivos y leer o escribir estos por los clientes realizando un control de las versiones almacenadas en él.

- **TortoiseSVN:** Gestor gráfico que permite la interacción con la información suministrada por el repositorio a través de una interfaz más intuitiva.

- **Fiddler:** Es un proxy HTTP virtual que captura las solicitudes realizadas por el navegador y permite inspeccionar en detalle la comunicación con el servidor web.

2.2.3 Aspectos teórico básicos

2.2.3.1 Ciclo de vida de una aplicación ASP.NET

Para conseguir los efectos deseados en las aplicaciones ASP.NET es importante conocer su ciclo de vida para poder incorporar el código necesario dentro de cada fase.

El ciclo de vida de una aplicación ASP.NET se inicia con una solicitud enviada por el navegador al servidor Web, para las aplicaciones ASP.NET, normalmente es IIS. ASP.NET es una extensión ISAPI bajo el servidor Web. Cuando un servidor Web recibe una solicitud, examina la extensión de nombre de archivo del archivo solicitado, determina la extensión ISAPI que debería procesar dicha solicitud y, a continuación, pasa ésta a la extensión ISAPI apropiada.

Cuando ASP.NET recibe la primera solicitud para cualquier recurso de una aplicación, se crea un dominio de aplicación. Los dominios de aplicación proporcionan aislamiento entre aplicaciones para las variables globales y permiten descargar cada aplicación de forma independiente. Dentro de un dominio de aplicación, se crea una instancia de la clase denominada `HostingEnvironment`, que proporciona acceso a la información sobre la aplicación, como el nombre de la carpeta en la que está almacenada la aplicación.En el diagrama siguiente se muestra esta relación.

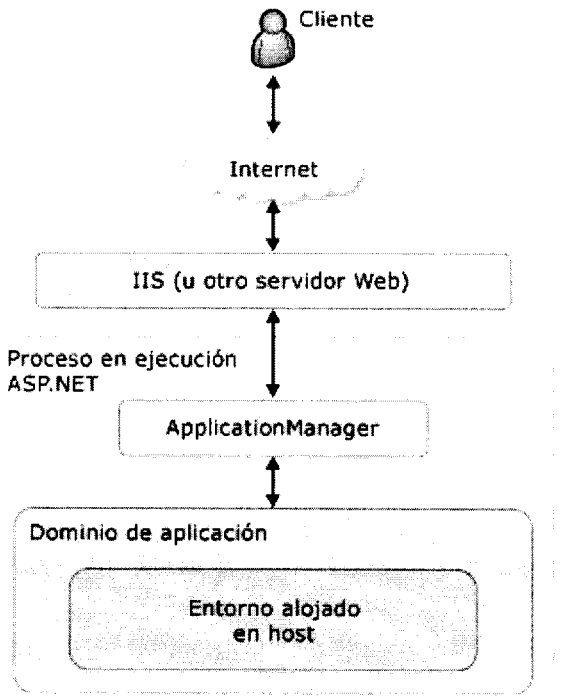


Figura 2.1 Ciclo de vida de una aplicación Web.

Una vez creados el dominio de aplicación, ASP.NET crea e inicializa objetos de núcleo, como `HttpContext`, `HttpRequest` y `HttpResponse`. La clase `HttpContext` contiene objetos específicos de la solicitud de aplicación actual, como los objetos `HttpRequest` y `HttpResponse`. El objeto `HttpRequest` contiene datos sobre la solicitud actual, entre los que se incluyen las cookies e información del explorador. El objeto `HttpResponse` contiene la respuesta que se envía al cliente, la cual incluye todos los resultados presentados y las cookies.

Una vez que se han inicializado todos los objetos principales de la aplicación, ésta se inicia creando una instancia de la clase `HttpApplication`. Si la aplicación tiene un archivo `Global.asax`, ASP.NET crea una instancia de la clase `Global.asax` derivada de la clase `HttpApplication` y la utiliza para representar la aplicación. Cuando se crea una instancia de `HttpApplication`, también se crean los módulos configurados.En el diagrama siguiente se muestra esta relación.

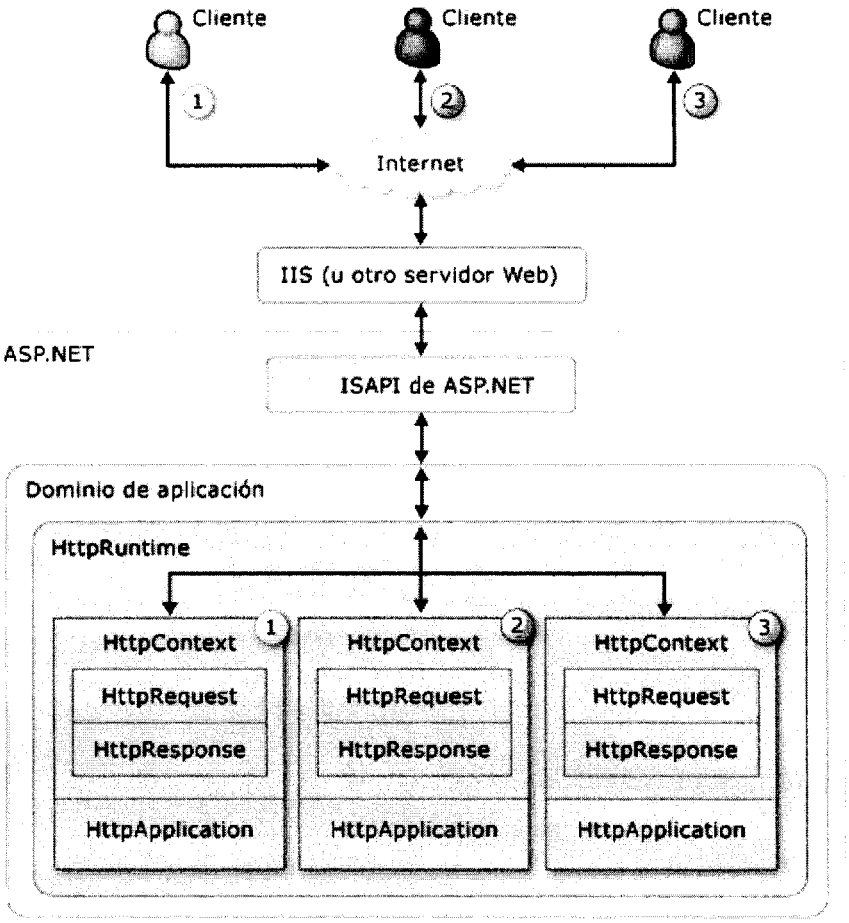


Figura 2.2 Ciclo de vida de una página ASP.NET.

2.2.3.2 Fases generales del ciclo de vida de una página en ASP.NET

Una página ASP.NET al ejecutarse recorre un ciclo de vida en el que realiza una serie de pasos de procesamiento. Entre ellos se incluyen la inicialización, la creación de instancias de controles, la restauración y el mantenimiento del estado, la ejecución del código del controlador de eventos y la representación.

Es importante el conocimiento del ciclo de vida de una página ASP.NET pues en él está basado el ciclo de vida de un control Web personalizado.

- **Solicitud de página:** Se produce antes de que comience el ciclo de vida de la página. Cuando un usuario solicita la página, ASP.NET determina si esta se debe analizar y

compilar o si se puede enviar una versión en caché de la página como respuesta sin ejecutar la página.

- **Inicio:** Se establecen las propiedades de la página, como `Request` y `Response`. En esta fase, la página también determina si la solicitud es una devolución de datos o una nueva solicitud, y establece la propiedad `IsPostBack`. Además, durante esta fase se establece la propiedad `UICulture` de la página. Si la solicitud es una devolución de datos, los valores de los controles todavía no se han restaurado del estado de vista. Durante esta fase se recomienda utilizar la propiedad `IsPostBack` para determinar si esta es la primera vez que se procesa la página y crear o volver a crear controles dinámicos.

  Evento: `Page_PreInit`.

- **Inicialización de página:** Durante la inicialización de la página, los controles incluidos en ella están disponibles y se establece la propiedad `UniqueID` de cada uno de ellos. Además, se aplican los temas correspondientes a la página. Si la solicitud actual es una devolución de datos, los datos de devolución aún no se han cargado y los valores de las propiedades del control no se han restaurado a los valores del estado de vista. Durante esta fase se recomienda leer o inicializar las propiedades de los controles.

  Evento: `Page_Init`.

- **Carga:** Durante la carga, si la solicitud actual es una devolución de datos, las propiedades del control se cargan con información recuperada del estado de vista y del estado del control. Durante esta fase se recomienda leer y actualizar las propiedades de los controles.

  Evento: `Page_Load`.

- **Validación:** Durante la validación, se llama al método `Validate` de todos los controles de validación, que establece la propiedad `IsValid` de cada uno de los controles de validación y de la página.

- **Control de eventos de devolución de datos:** Si la solicitud es una devolución de datos, se llama a los controladores de eventos.

- **Eventos:** Se ejecutan los eventos definidos en cada control.

- **Representación:** Durante la representación, el estado de vista se guarda en la página y a continuación esta llama a cada uno de los controles para que aporte su salida representada al valor `OutputStream` de la propiedad `Response` de la página. Se recomienda durante esta fase realizar los cambios finales en el contenido de la página.

  Evento: `Page_PreRender`.

- **Descarga:** Se llama a la descarga cuando la página se ha representado completamente, se ha enviado al cliente y está lista para ser descartada. Llegado este momento, se descargan las propiedades de la página, como `Response` y `Request`, y se realizan las operaciones de limpieza correspondientes. Durante esta fase se deben cerrar los archivos abiertos y las conexiones a bases de datos así como la finalización del registro o de otras tareas específicas de cada solicitud.

  Evento: `Page_Unload`.

Otros aspectos a considerar

- Cada uno de los controles de servidor ASP.NET tiene su propio ciclo de vida, que es similar al ciclo de vida de la página. Se llama a los métodos `Init` y `Load` de un control durante los eventos de página correspondientes. Si se ha incluido un control en la página, se llamará primero al método `Init` de dicho control, seguido por el método `Init` de la página. Sin embargo, se llamará antes al método `Load` de la página que al método `Load` del control.

- Para personalizar la apariencia o el contenido de un control, se debe controlar sus eventos.

- Se puede reemplazar los métodos de la clase base de la página. Si reemplaza un método de la página se debe llamar de forma explícita al método base para poder ejecutar la implementación base.

2.2.3.3 Creación de controles en ASP.NET

ASP.NET brinda la posibilidad de ampliar su librería de controles para satisfacer las necesidades del usuario que no son cubiertas por los controles estándares que se encuentran en la

plataforma. Existen dos maneras fundamentales para su creación en función de los requerimientos de trabajo a desarrollar y del grado de conocimiento del programador.

Controles Web de usuario

Son de fácil creación pues se desarrollan en el diseñador visual al igual que un formulario Web, manteniendo por separado la interfaz del código. Sin embargo como los controles Web de usuarios son compilados en tiempo de ejecución, no se pueden adicionar a la barra de herramientas y son representados por un *placeholder* al ser adicionados a la página. Esto limita las posibilidades de configuración en tiempo de diseño brindada por el marco de trabajo .NET y también hace que la única vía para compartir el control entre diferentes páginas de la aplicación sea copiar el código por separado en cada una de ellas. Recomendado para la creación de controles que no modifiquen dinámicamente su interfaz.

Controles Web personalizados

Son código compilado por lo que para su desarrollo necesita de un mayor nivel de conocimientos acerca de la arquitectura de los controles y su ciclo de vida, una vez creado puede ser adicionado a la barra de herramientas y ser arrastrado y mostrado en formulario Web o bien incluidos en la página registrándolo en el ensamblado global de la aplicación. Brindan todas las posibilidades de creación y configuración en tiempo de diseño que soportan los controles Web de .NET, como plantillas, diseñadores y convertidores de tipo, lo que facilita su personalización de manera gráfica. Son la mejor opción cuando se van a crear controles dinámicos, por ejemplo los controles enlazados a datos.

Existen tres vías para la creación de un control Web personalizado.

- Derivar de un control Web existente para ampliar o redefinir sus funcionalidades. Útil cuando se quiere modificar parcialmente la lógica de un control existente por ejemplo para incluir funciones script para la validación en el cliente del control.

- Heredar de la clase base control e implementar completamente la interfaz y las funcionalidades del control.

- Crear un control compuesto que utilice controles secundarios para crear una interfaz de usuario e implementar otro tipo de lógica. Es mucho más fácil crear un control compuesto que implementar toda la funcionalidad del control con los medios propios.

2.2.3.4 Arquitectura de un control Web en tiempo de diseño

.NET Framework ofrece interfaces y clases para personalizar el comportamiento de componentes y las interfaces de usuario en un entorno en tiempo de diseño. Normalmente, un entorno en tiempo de diseño incluye un diseñador de formularios para organizar componentes y un examinador de propiedades para configurar los valores de las propiedades de un componente. Un entorno en tiempo de diseño, normalmente, también proporciona servicios en tiempo de diseño a los que se puede obtener acceso y que se pueden utilizar mediante mecanismos en tiempo de diseño.

También como parte del marco de trabajo se definen interfaces que los programadores pueden utilizar para conseguir funcionalidad personalizada de los componentes en tiempo de diseño. Los mecanismos principales de extensión de la funcionalidad en tiempo de diseño se incluyen en las categorías siguientes: diseñadores, convertidores de tipos y editores de tipos de la interfaz de usuario. Los atributos se aplican a tipos y a miembros de tipo que los asocian con estos proveedores de funcionalidad en tiempo de diseño.

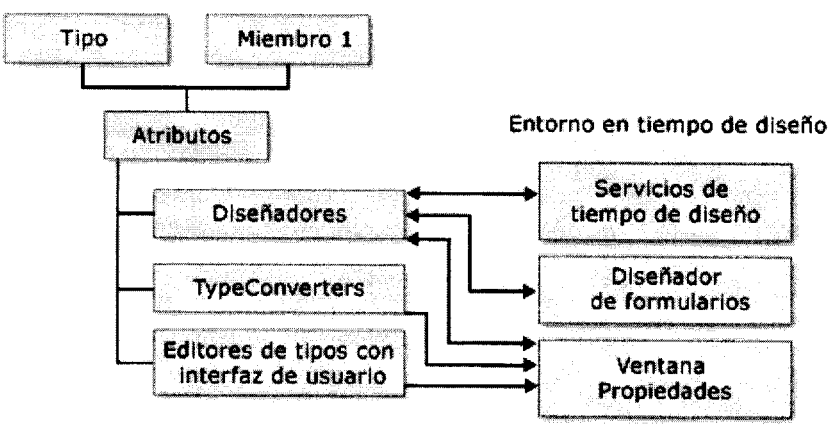


Figura 2.3 Arquitectura de un control Web en tiempo de diseño.

Atributos

Los atributos son etiquetas descriptivas que proporcionan información adicional sobre elementos de programación como tipos, campos, métodos y propiedades. Constituyen un medio apropiado para asociar información declarativa con código. (10) Pueden extender el comportamiento en tiempo de diseño. `DesignerAttribute` asocia un diseñador a una clase. `TypeConverterAttribute` asocia un tipo o un miembro de un tipo con un convertidor de tipos.

`EditorAttribute` asocia un tipo o un miembro de un tipo con un editor de tipos de interfaz de usuario.

Diseñadores de controles

"Un diseñador de controles permite la representación en tiempo de diseño de un control de servidor Web actuando como mediador entre el control en tiempo de ejecución y el entorno host" (11), incluyendo la apariencia, inicialización e interacción con el usuario. Un diseñador puede agregar, eliminar o reemplazar propiedades enumeradas en un examinador de propiedades de un componente seleccionado. Un diseñador puede proporcionar métodos definidos por el usuario que se pueden vincular a ciertos eventos de componente o ejecutar desde un comando de menú personalizado o desde `DesignerVerb`. Un diseñador también puede utilizar servicios proporcionados por un entorno en tiempo de diseño.

Convertidores de tipos

Se puede implementar un convertidor de tipos para convertir valores entre el tipo de datos al que da servicio y otros tipos de datos de los y a los que puede trasladar valores. Un convertidor de tipos también puede proporcionar una lógica que permita la configuración de una propiedad en un examinador de propiedades en tiempo de diseño. Un convertidor de tipos puede proporcionar una lista de valores estándar para una propiedad del tipo de datos al que da servicio en tiempo de diseño en un examinador de propiedades. Un convertidor de tipos también puede generar código de inicialización para inicializar una propiedad en tiempo de diseño.

Editores de tipos con interfaz de usuario

Un editor de tipos de interfaz de usuario puede proporcionar una interfaz de usuario personalizada para modificar el valor de la propiedad y mostrar una representación del valor de la propiedad en tiempo de diseño. Un editor con tipos de interfaz de usuario es específico de tipos y proporciona una interfaz de usuario para configurar propiedades de los tipos a los que da servicio, o tipos derivados que no tienen un atributo que se reemplaza, en tiempo de diseño. Un editor de tipos con interfaz de usuario puede mostrar un formulario Windows o una interfaz de configuración de menú desplegable para configurar una propiedad.

Tipos relacionados con el diseño

A continuación se muestran algunas clases importantes en los espacios de nombres relacionados con el diseño.

`System.Drawing.Design`

- `UITypeEditor`: Proporciona una clase base que se puede utilizar para implementar editores de valores.

`System.ComponentModel.Design`

- `DesignerActionList`: Proporciona la clase base para los tipos que definen una lista de elementos utilizada para crear un panel de etiquetas inteligentes.
- `IDesigner`: Proporciona el marco básico para generar un diseñador personalizado.
- `IRootDesigner`: Proporciona funcionalidad para las tecnologías de vista de diseñador de nivel raíz.

`System.Windows.Forms.Design`

- `IWindowsFormsEditorService`: Proporciona una interfaz para que los editores de tipos de interfaz de usuario puedan mostrar formularios o un control en un área desplegable del panel de propiedades en modo de diseño.
- `ControlDesigner`: Clase de diseñador base para ampliar el comportamiento en modo de diseño de Control.

Servicios en tiempo de diseño

.NET Framework proporciona un conjunto de servicios en tiempo de diseño que pueden extender las capacidades de un diseñador. Estos servicios se pueden obtener utilizando el método `GetService` de un componente ubicado en modo de diseño. Se pueden agregar tipos de servicio propios a los que se puede obtener acceso desde un proyecto de modo de diseño mediante el método `AddService` de la interfaz de servicio `IDesignerHost`.

La interfaz `IComponentChangeService` permite al diseñador recibir notificaciones sobre cuándo se cambian, se agregan o se eliminan componentes del entorno en tiempo de diseño.

La interfaz `IDesignerEventService` permite a un diseñador recibir notificaciones cuando se agregan o eliminan diseñadores del entorno en tiempo de diseño, así como cuando cambia la selección del componente actual.

La interfaz `IDesignerFilter` permite al diseñador agregar y filtrar el conjunto de propiedades que se muestran en un examinador de propiedades de su componente.

La interfaz `IDesignerHost` proporciona una interfaz para agregar y recuperar servicios, controlar eventos relacionados con el estado del diseñador, detectar si un diseñador se está cargando en ese momento y administrar componentes o transacciones del diseñador. Las transacciones del diseñador permiten que se produzcan secuencias de acciones que impidan que la vista en tiempo de diseño actualice la pantalla hasta que finalice una secuencia de acciones para mejorar el rendimiento, mientras que suministran un mecanismo que permite que se deshagan las acciones del componente de una transacción y que se restaure el estado anterior.

La interfaz `IRootDesigner` permite a un diseñador reemplazar la vista principal del diseño con una presentación personalizada.

2.3 Fase de planificación de entregas

2.3.1 Historia de usuarios

Las historias de usuarios es la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado. El cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Se propone utilizar un nombre, usuario, fecha de inicio y fecha final, la prioridad, la estimación en puntos y la descripción para hacer un seguimiento de cada historia.

Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, se mantiene un registro de la velocidad

de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación. A continuación cada una de las historias de usuarios definida.

Historia de usuario		
Número: 1	**Nombre:** Definir interfaz.	
Usuario: MININT	**Fecha de inicio:** 04/02/2008	**Fecha de fin:** 15/02/2008
Prioridad: Alta	**Puntos: 1**	
Descripción: Se define la interfaz de manera gráfica, utilizando para ello los controles que brinda el Visual Studio 2005 para ASP.NET. De esta forma el usuario puede ser capaz de diseñar y modificar la estructura visual que será representada por el control en la página Web.		

Tabla 2.1 Historia de usuario: Definir interfaz.

Historia de usuario		
Número: 2	**Nombre:** Definir fuente de datos.	
Usuario: MININT	**Fecha de inicio:** 16/02/2008	**Fecha de fin:** 03/03/2008
Prioridad: Alta	**Puntos:** 2	

Descripción:
El usuario debe ser capaz de definir la fuente de datos a utilizar y que la misma pueda ser vinculada con la interfaz visual que el usuario defina. Esta fuente de datos debe tener las mismas características que las que utilizan los controles enlazados a datos en ASP.NET.

Tabla 2.2 Historia de usuario: Definir fuente de datos.

Historia de usuario		
Número: 3	**Nombre:** Editar los datos representado por los controles.	
Usuario: MININT	**Fecha de inicio:** 04/03/2008	**Fecha de fin:** 24/03/2008
Prioridad: Media	**Puntos:** 2	
Descripción: El usuario debe poder, en dependencia de la fuente de datos definida, insertar, modificar o eliminar cualquiera de dichos datos. Todo esto en tiempo de ejecución. Para ello debe definir los identificadores necesarios para realizar la operación deseada.		

Tabla 2.3 Historia de usuario: Editar los datos representados por los controles.

Historia de usuario		
Número: 4	**Nombre:** Definir estilos.	
Usuario: MININT	**Fecha de inicio:** 25/03/2008	**Fecha de fin:** 02/04/2008
Prioridad: Baja	**Puntos:** 1	
Descripción: El usuario define el estilo visual de cada objeto que genere el control, pudiendo modificar uno o todos los objetos. Además debe poder especificarle a cada objeto una clase de estilo css, la cual él tendría definida previamente.		

Tabla 2.4 Historia de usuario: Definir estilos.

Historia de usuario		
Número: 5	**Nombre:** Crear librería en JavaScript.	
Usuario: MININT	**Fecha de inicio:** 04/02/2008	**Fecha de fin:** 19/02/2008
Prioridad: Alta	**Puntos:** 2	
Descripción: El componente debe tener agregado una librería en JavaScript para manejar las funcionalidades en el cliente. De esta forma se pueden realizar operaciones con los elementos representados en el control pero en tiempo de ejecución. Esta librería funcionaría como respaldo en el cliente para mantener el control en ASP.NET.		

Tabla 2.5 Historia de usuario: Crear librería en JavaScript.

Historia de usuario		
Número: 6	**Nombre:** Comportamiento sincrónico y asincrónico del control.	
Usuario: MININT	**Fecha de inicio:** 25/03/2008	**Fecha de fin:** 02/04/2008
Prioridad: Media	**Puntos:** 1	
Descripción: El control debe manejar los datos de forma sincrónica o asincrónica. Para el trabajo asincrónico debe definir sus propias funciones o debe ser compatible con controles ASP.NET Ajax que ya lo tengan incluido. Ejemplo: UpdatePanel		

Tabla 2.6 Historia de usuario: Comportamiento sincrónico y asincrónico del control.

Historia de usuario		
Número: 7	**Nombre:** Mover los objetos del control en pantalla.	
Usuario: MININT	**Fecha de inicio:** 04/03/2008	**Fecha de fin:** 24/03/2008
Prioridad: Media	**Puntos:** 2	
Descripción: La librería JavaScript agregada al control debe tener las funcionalidades necesarias para que los elementos se puedan mover o no en la pantalla (utilizando el mouse). Para esto el usuario puede definir cuando y como quiere realizar este movimiento. Esta funcionalidad debe ser compatible con los navegadores más usados actualmente.		

Tabla 2.7 Historia de usuario: Mover los objetos del control en pantalla.

Historia de usuario		
Número: 8	**Nombre:** Dimensión, posición y opacidad de los elementos.	
Usuario: MININT	**Fecha de inicio:** 20/02/2008	**Fecha de fin:** 03/03/2008
Prioridad: Baja	**Puntos:** 1	
Descripción: La librería JavaScript agregada al control debe tener las funcionalidades necesarias para que los elementos se puedan cambiar su tamaño, posición y opacidad en tiempo de ejecución.		

Tabla 2.8 Historia de usuario: Dimensión y opacidad de los elementos.

Historia de usuario		
Número: 9	**Nombre:** Convertir objeto ASP.NET a objeto JavaScript y viceversa.	
Usuario: MININT	**Fecha de inicio:** 24/04/2008	**Fecha de fin:** 02/05/2008
Prioridad: Alta	**Puntos:** 1	
Descripción: El objeto ASP.NET que representa al control, el cual contiene todos los elementos representados debe convertirse en un objeto JavaScript que pasaría a ser manejado desde la librería JavaScript incluida dentro del control para permitir que el usuario pueda manipular o modificar las principales propiedades de los elementos representados en pantalla.		

Tabla 2.9 Historia de usuario: Convertir objeto ASP.NET a objeto JavaScript y viceversa.

Historia de usuario		
Número: 10	**Nombre:** Permitir la paginación de los datos.	
Usuario: MININT	**Fecha de inicio:** 24/04/2008	**Fecha de fin:** 02/05/2008
Prioridad: baja	**Puntos:** 1	
Descripción: El control debe brindar la posibilidad al usuario de definir si se han de paginar los datos y el tamaño de página.		

Tabla 2.10 Historia de usuario: Permitir la paginación de datos.

Historia de usuario		
Número: 11	**Nombre:** Persistir datos de estado.	
Usuario: MININT	**Fecha de inicio:** 03/04/2008	**Fecha de fin:** 23/04/2008
Prioridad: media	**Puntos:** 2	
Descripción: El usuario puede definir que propiedades o características de configuración del control deben persistir cuando se modifiquen sus valores.		

Tabla 2.11 Historia de usuario: Persistir datos de estado.

Historia de usuario		
Número: 12	**Nombre:** Identificar sesión de diseñador.	
Usuario: MININT	**Fecha de inicio:** 03/04/2008	**Fecha de fin:** 23/04/2008
Prioridad: media	**Puntos:**2	
Descripción: El control debe ser capaz de reconocer cual es la sesión de diseño actual y configurarse de acuerdo a las últimas características persistidas.		

Tabla 2.12 Historia de usuario: Identificar sesión de diseñador.

2.3.2 Prototipo del control

Se define un esquema de la arquitectura base del control que contará de cinco módulos donde se agrupan las clases a desarrollar.

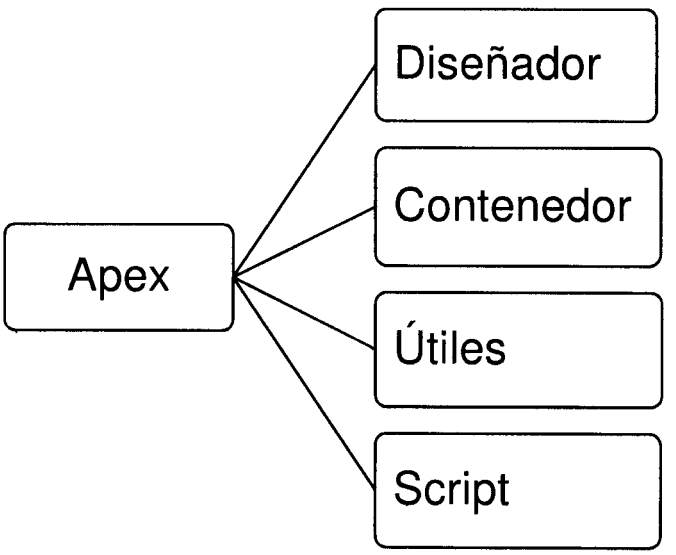


Figura 2.4 Prototipo de la arquitectura del control.

- **Apex:** Lo constituye la clase del control.
- **Diseñador:** Agrupa todas las clases encargadas de la configuración en tiempo de diseño del control, estas clases pueden ser diseñadores, convertidores o editores de tipo de acuerdo a la arquitectura planteada para un control Web en .NET.
- **Contenedor:** Forman partes de él las clases donde se instancian los ítem creados dinámicamente por el control.
- **Script:** Contiene el conjunto de librerías de clases script que le dan soporte al control en el cliente.

2.3.3 Planificación de cada entrega

La prioridad ha sido especificada de acuerdo a los intereses del usuario y con vista a maximizar el valor del producto en el menor tiempo. De acuerdo a esto se ha elaborados el plan de entrega para cada iteración utilizando el método de planificación según el alcance del sistema descrito anteriormente en el documento, procurando agrupar las funcionalidades relacionadas en la misma iteración.

Historia de usuario	Primera Iteración	Segunda iteración	Tercera iteración
Definir interfaz.	1		
Definir fuente de datos.	1		
Dimensión, posición y opacidad de los elementos.	1		
Crear librería en JavaScript.	1		
Definir estilos.		1	
Comportamiento sincrónico y asincrónico del control.		1	
Mover los objetos del control en pantalla.		1	
Editar los datos representado por los controles.		1	
Convertir objeto ASP.NET a objeto JavaScript y viceversa.			1
Permitir la paginación de los datos.			1
Persistir datos de estado.			1
Identificar sesión de diseñador.			1

Tabla 2.13 Plan de entrega por iteraciones.

2.3.3.1 Primera iteración

Durante esta iteración se implementará una primera versión de control que implemente las funcionalidades generales además de aquellas vinculadas con la interfaz gráfica en tiempo de diseño, la aceptación de datos y su representación.

Esfuerzo estimado

Funcionalidad	Puntos
Definir interfaz.	1
Definir fuente de datos.	2
Crear librería en JavaScript.	2
Dimensión, posición y opacidad de los elementos.	1
Total	**6**

Tabla 2.13 Esfuerzo estimado en la iteración 1.

2.3.3.2 Segunda iteración

Esta iteración abarcará la implementación de funcionalidades que permitan la interacción gráfica con el usuario basando su mayor peso en las funcionalidades del cliente y la refactorización de las ya implementadas.

Esfuerzo estimado

Funcionalidad	Puntos
Editar los datos representado por los controles.	2
Definir estilos.	1
Comportamiento sincrónico y asincrónico del control.	1
Mover los objetos del control en pantalla.	2
Total	**6**

Tabla 2.14 Esfuerzo estimado en la iteración 2.

2.3.3.3 Tercera iteración

En la última iteración se abarcarán las funciones para permitir la persistencia de datos.

Esfuerzo estimado

Funcionalidad	Puntos
Convertir objeto ASP.NET a objeto JavaScript y viceversa.	1
Permitir la paginación de los datos.	1
Persistir datos de estado.	2
Identificar sesión de diseñador.	2
Total	**6**

Tabla 2.15 Esfuerzo estimado en la iteración 3.

2.4 Conclusiones

Al finalizar estas dos fases se ha hecho un estudio de los aspectos necesarios para comenzar a desarrollar el control, se han definido cada una de las funcionalidades requeridas por el usuario y se ha planteado un cronograma de entrega por iteraciones en función del alcance del sistema. Estos dos últimos aspectos pueden variar en la próxima fase de desarrollo de la metodología de acuerdo a los principios y prácticas de XP.

CAPÍTULO 3: APLICACIÓN DE LA FASE DE ITERACIONES

3.1 Introducción

Durante este capítulo se hace un análisis de cada una de las iteraciones realizadas sobre el sistema en concordancia con el plan de entrega definido en el capítulo anterior. Cada una de ellas con una duración máxima de tres semanas con el objetivo de disminuir los plazos de entrega y maximizar el valor del producto. Al principio de cada iteración se tendrá en cuenta, las historias de usuarios no terminadas, las no abordadas y los posibles cambios tanto en la arquitectura como en el código para la nueva iteración.

3.2 Fase de Iteración

3.2.1 Primera Iteración

Es el comienzo del desarrollo del software, en esta etapa se desarrollan las funcionalidades básicas del control que propicien una implementación base de su arquitectura que de cómo resultado un producto funcional. Se abordan las historias de usuario:

- Definir interfaz.
- Definir fuente de datos.
- Crear librería en JavaScript.
- Dimensión, posición y opacidad de los elementos.

3.2.2 Definir interfaz

Esta historia de usuario tiene como inicio la necesidad de permitir a los desarrolladores definir la interfaz gráfica del control de manera personalizada. Para su implementación se divide en dos tareas fundamentales, la definición de un control Web personalizado y la inclusión de propiedades de plantillas como medio para proveer de interfaz gráfica a los controles. La arquitectura generada consta de tres clases: `Apex`, `Contenedor` y `ApexDisenador`.

Nuevas clases

- **Apex:** Constituye la clase del control a representar y sobre ella se centra cada una de las iteraciones a realizar. Se extiende de la clase `CompositeControl` definida en el marco de trabajo de .NET y que brinda funcionalidades de contenedor de nomenclatura y diseñador de controles para controles personalizados que contienen controles secundarios en su totalidad.

Los métodos y propiedades más importantes de esta clase son:

  - `CreateChildControls`: En él se genera la estructura del control creando instancias de las plantillas definidas por el usuario.

  - `ITemplate`: Las propiedades de tipo `ITemplate` son utilizadas para definir las plantillas utilizadas por el control para gestionar los datos.

  - `TagKey`: Propiedad de la clase base que define la etiqueta HTML a utilizar para renderear el control en las páginas Web. Se utiliza para ello la etiqueta `DIV` por su flexibilidad en la Web como contenedor de controles.

- **ApexDisenador:** Define el comportamiento en tiempo de diseño del control Apex, hereda de la clase `ControlDesigner` que ofrece facilidades de trabajo para la creación de formatos de estilo, la creación de listas de acciones o etiquetas inteligentes y la definición de plantillas. La propiedad más importante es `TemplateGroups` que sobrescribe la existente en la clase base para definir en ella la colección de plantillas a mostrar por el control. Durante esta historia de usuario se define la plantilla: LecturaPlantilla.

- **Contenedor:** Clase donde se crean las instancias de cada plantilla antes de ser adicionadas al control Apex. Hereda de la clase panel para facilitar la compatibilidad entre los objetos del cliente y el servidor.

3.2.2.1 Definir fuente de datos

Se apoya en las características definidas en .NET para la utilización de controles de orígenes de datos y de controles enlazados a datos. Para lograr cumplir este requisito se hace una refactorización de la arquitectura anterior y se extiende la clase `Apex` de

`CompositeDataBoundControl`, la clase `ApexDisenador` de `DataboundControlDesigner` y `Contenedor` de la interfaz `IDataItemContainer`. A continuación una breve explicación de cada una de estas clases:

- **`CompositeDataBoundControl`:** Clase que forma parte del marco de trabajo .NET de donde extienden los controles enlazados a datos declarando un nuevo método `CreateChildControls`.

- **`DataboundControlDesigner`:** Clase perteneciente al marco de trabajo .NET que proporciona la interfaz necesaria en tiempo de diseño para definir una fuente de datos a la cual se enlaza el control al que pertenece.

- **`IDataItemContainer`:** Interfaz de trabajo perteneciente al marco de trabajo .NET que define las propiedades necesarias a implementar para los contenedores de controles enlazados a datos.

Refactorización

- **`Apex`:** Se modifica el método `CreateChildControls` para crear una instancia de la plantilla con la que se han de mostrar los datos por cada tupla definida en la fuente de datos del control.

3.2.2.2 Crear librería en JavaScript

Debido a que todas las características y funcionalidades del control se manejan del lado del servidor, es necesaria la creación de una librería para dar soporte al control del lado del cliente y así aparte de ampliar su funcionalidad, lograr una mayor rapidez en tareas que no son necesarias entregar al servidor. Con este principio se crea una librería en JavaScript, la cual presenta un arquitectura de tres clases: `Apex`, `Elemento` y `Error`. Es importante aclarar que aunque JavaScript no es un lenguaje orientado a objetos totalmente y el uso del mismo generalmente es a través de funciones o scripts incluidos en las páginas Web, se ha logrado un mecanismo para utilizarlo de forma orientada a objeto. Para la creación de este mecanismo, fundamentalmente con relación a la organización en `namespace` y clases, ha sido utilizada la librería ASP.NET Ajax de Microsoft, la cual, entre otros aspectos importante, se encarga de proporcionar funcionalidades para simular en JavaScript el comportamiento del Framework.Net.

Nuevas clases

- **Apex:** Es la clase principal de la librería, a través de ella se accede a todas las funcionalidades del lado del cliente que permita la librería. Cada una de estas funcionalidades se encuentran organizadas en distintas clases y las propiedades y métodos de estas a su vez son llamados a través de objetos de las mismas desde la clase `Apex`. Este clase `Apex` contiene varios atributos que representan objetos de las clases `Elemento` y `Error` en cada uno de los casos.

- **Error:** Clase encargada de gestionar todos los posibles errores que se produzcan en la librería. A través de la clase `Apex` es posible hacer un llamado a las propiedades y métodos de la misma. No obstante para enriquecer más el DOM de los distintos navegadores, una vez incluida esta librería todas las propiedades y métodos de esta clase pasan a formar parte también de la clase `Error` del DOM, por lo que es posible acceder a ellos directamente sin necesidad de hacerlo a través del objeto de esta clase que se encuentra dentro de la clase `Apex`.

- **Elemento:** Esta clase es la encargada de gestionar los elementos que son incluidos dentro de la colección que maneja la librería, estos elementos se incluirán automáticamente una vez configurado el control y ejecutado la página o manualmente se pueden incluir cualquier objeto HTML. Esta clase presenta propiedades y métodos que ayudan a interactuar con dichos elementos.

Las propiedades y métodos más importantes son:

- **agregar:** Método que se encarga de agregar un nuevo elemento a la colección de la librería. De este elemento solo se necesita especificar su id para ser agregado. Una vez adicionado el elemento se pueden utilizar con él todas las funcionalidades de la librería.

- **eliminar:** Método que elimina un elemento que se encuentre en la colección de elementos. De este elemento solo se necesita especificar su id para ser eliminado.

- **obtener:** Método que devuelve un elemento que se encuentre en una posición determinada en la colección. Para lograr esto se debe especificar el índice de la posición.

- **coleccion:** Propiedad que representa a la colección de elementos que se encuentran registrados en la librería. Esta propiedad es una arreglo asociativo, por lo que se puede acceder a un elemento en determinado especificado su id como índice del arreglo.

3.2.2.3 Dimensión, posición y opacidad de los elementos

Con el objetivo de agregarle mayor funcionalidad a la librería de JavaScript se han agregado los métodos `posicion`, `dimension` y `opacidad`, estos métodos pertenecen a la clase `Elemento`. De esta forma es posible cambiar o adicionar estos valores a los elementos de la colección en tiempo de ejecución y con sintaxis sencilla. En el caso particular de la opacidad se tiene en cuenta las incompatibilidades que existen entre algunos navegadores Web para realizar esta tarea, logrando una forma común de hacerlo aunque internamente se tenga en cuenta el aspecto de las incompatibilidades entre los navegadores Web.

3.2.3 Segunda Iteración

Durante esta iteración se amplían las funcionalidades abordadas anteriormente y se adicionan otras. El trabajo se centra en permitir la gestión de los datos mostrados y el tratamiento gráfico de los mismos en tiempo de ejecución. Las historias de usuarios definidas en el plan de entrega para esta etapa son:

- Editar los datos representado por los controles.
- Definir estilos.
- Comportamiento sincrónico y asincrónico del control.
- Mover los objetos del control en pantalla.

3.2.3.1 Editar los datos representado por los controles

Esta historia de usuario da soporte a todas las funciones relacionadas con la gestión de los datos. En ella se definen nuevas plantillas de diseño en el control y se implementa además como responder a los eventos de los controles que contienen de acuerdo a la propiedad `CommandArgument` definida. Cada ítem del control puede estar en tres Modos: `Edicion`, `Insercion` y `Lectura` siendo este el valor por defecto. `Apex` solo puede contar con un ítem en

modo de Edición y uno en modo de Inserción. Para la configuración de cada acción se sigue el ciclo expuesto a continuación.

- Eliminar: Se le aplica este CommandArgument a un botón o control con dicha propiedad contenido en la plantilla Lectura.
- Insertar: Se le aplica este CommandArgument a un botón o control con dicha propiedad contenido en la plantilla Insertar.
- Actualizar: Se le aplica este CommandArgument a un botón o control con dicha propiedad contenido en la plantilla Actualizar.
- Nuevo: Se le aplica este CommandArgument a un botón o control con dicha propiedad contenido en la plantilla lectura para que el ítem contenido cambie al modo de inserción.
- Editar: Se le aplica este CommandArgument a un botón o control con dicha propiedad contenido en la plantilla lectura para que el ítem contenido cambie al modo de edición.
- Seleccionar: Se le aplica este CommandArgument a un botón o control con dicha propiedad contenido en la plantilla lectura.

Nuevos métodos

- **OnBubbleEvent:** Perteneciente a la clase Apex. Captura los eventos lanzados en los controles hijos para tratarlos en función del argumento pasado como parámetro. Para cada tipo de argumento existe un manejador específico encargado de ejecutar la acción.

Eventos

El control define eventos que permiten un mayor control de su funcionamiento por parte del usuario durante el ciclo de vida del control. A continuación una tabla con los eventos específicos del control y su función.

Evento	Argumento	Descripción
OnItemCreated	ContenedorEventArgs	Se lanza cuando se crea un nuevo ítem.

OnItemDataBound	ContenedorEventArgs	Se lanza cuando se enlaza un ítem a datos.
OnSelectedIndexChanged	EventArgs	Se lanza cuando cambio el índice del ítem seleccionado.
OnSelectedIndexChanging	ApexSelectEventArgs	Se lanza antes de cambiar el índice seleccionado.
OnRowEditing	ApexEditEventArgs	Se lanza antes de que el ítem entre en modo de edición.
OnRowDeleted	ApexDeletedEventArgs	Se lanza cuando se ha borrado un ítem.
OnRowDeleting	ApexDeleteEventArgs	Se lanza antes de borrar el ítem.
OnRowInsertado	ApexInsertadoEventArgs	Se lanza cuando se ha insertado un nuevo ítem.
OnRowInsertando	ApexInsertarEventArgs	Se lanza antes de insertar un nuevo ítem.
OnRowUpdating	ApexUpdateEventArgs	Se lanza antes de actualizar un ítem.
OnRowUpdated	ApexUpdatedEventArgs	Se lanza se lanza cuando se ha actualizado un ítem.
OnRowCancelingEdit	ApexCancelEditEventArgs	Se lanza cuando se cancela el modo de edición.
OnRowCancelingInsert	ApexCancelInsertEventArgs	Se lanza cuando se cancela el modo de inserción.
OnRowNuevo	ApexNuevoEventArgs	Se lanza cuando se entra en modo de inserción.

Tabla 3.1 Controladores de eventos en Apex.

- **OnBubbleEvent:** Perteneciente a la clase Contenedor. Captura los eventos hijos y en función de su argumento es tratado por esta clase o lanzados a la clase padre para ser tratados.

Refactorizaciones

- Apex: La ampliación de las funciones del control implica un cambio en la manera de crear la jerarquía de controles. Se modifica para ello el método CreateChildControls donde a diferencia de la iteración anterior, por cada ítem generado se realiza una instancia de la plantilla en dependencia de la función a realizar.

- ApexDisenador: La colección de plantillas que representa la propiedad TemplateGroups se adicionan las platillas siguientes:

  - EditarPlantilla
  - InsertarPlantilla
  - VaciaPlantilla: Utilizada cuando la fuente de datos está vacía.

3.2.3.2 Definir estilos

La aplicación de estilos a cada ítem del control permite la personalización de su diseño mediante atributos Style o clases de estilos CSS.

Nueva propiedad

- **EstiloItem:** Perteneciente a la clase Apex. Es una propiedad de tipo ControlStyle que define el estilo a aplicar a cada ítem generado en Apex.

Nuevo método

- **ConvertirEstado:** Perteneciente a la clase Apex. Transforma los valores de los estilos de un control de servidor a una cadena de estilo para un control HTML.

Refactorización

- **Apex:** Se modifica el método `CreateChildControls` para permitir asignar los valores de la propiedad `EstiloItem` a los estilos de los ítems generados.

3.2.3.3 Comportamiento sincrónico y asincrónico del control

En la actualidad el desarrollo Web está centrado en el trabajo asincrónico, incorporar estas características al comportamiento del componente es uno de los requerimientos principales para su creación. Microsoft .NET Ajax contiene dos controles necesarios para realización de esta historia de usuario, ScriptManager y UpdatePanel.

El control ScriptManager es utilizado para administrar todas las instancias de los scripts generado por la librería del control por lo que es un requisito indispensable para el correcto funcionamiento de Apex su inclusión en la página.

Para definir el trabajo asincrónico se define Un UpdatePanel y dentro de él se crea la instancia del control y su fuente de datos.

3.2.3.4 Mover los objetos del control en pantalla

Para lograr que los elementos representados a través del control tengan mayor funcionalidad e interactividad con el usuario, se ha definido un mecanismo que permite mover los elementos por la pantalla en tiempo de ejecución, la librería en JavaScript que da soporte a las funcionalidades del lado del cliente al control es la encargada de llevar a cabo esta tarea. Además es posible suministrarle movilidad en pantalla en tiempo de ejecución a otros elementos que se desee y que deben agregarse previamente en la colección de elementos que maneja la librería. La ventaja fundamental de utilizar esta funcionalidad con los elementos representados a través del control es la posibilidad de hacer mediante una interfaz visual que permite asignarle movimiento a los elementos. Esto facilita el uso de la librería debido a que no es necesario tener conocimientos del lenguaje JavaScript para utilizar esta función sino que visualmente mediante la interfaz gráfica que brinda Visual Studio.NET 2005 y localizado específicamente en el panel de propiedades del control Apex.

Nuevas clases

- **Fabrica:** Se crea con el objetivo de organizar y centralizar más las funcionalidades de cada elemento almacenado en la colección. De esta forma en la colección almacenada lo

que se guarda a partir de la creación de la clase `Fabrica` es una instancia de la misma. Uno de los métodos de esta clase `Fabrica` es `mover`, el cual le asigna al elemento deseado la funcionalidad de moverse en pantalla en tiempo de ejecución.

Las propiedades y métodos más importantes son:

- `mover`: Método que asigna la funcionalidad de mover el objeto por la pantalla, logrando así una mayor interactividad con el usuario.

- `detener`: Método que elimina la funcionalidad de movimiento adicionada previamente al elemento, esto no quiere decir que elimine al elemento de la colección.

- `tipoMovimiento`: Propiedad que especifica el tipo de movimiento que va a realizar el elemento al que se le asigne dicha funcionalidad. De esta forma el elemento puede moverse horizontalmente, verticalmente o hacia todas direcciones, siendo esta última la opción por defecto.

- **`Holder`**: Con la asignación de movimiento a un elemento en pantalla que sea un componente visual capaz de ejecutar una petición al servidor, ocurre el problema de que al soltar dicho componente para dejar de efectuar el movimiento, se ejecuta una petición al servidor debido a todos los eventos necesarios para realizar esta petición se ejecutan también cuando se realiza el movimiento (`onmousedown`, `onmousemove` y `onmouseup`. Para evitar este comportamiento específico en dichos componentes y lograr que la funcionalidad de mover un elemento se efectúe sin problemas en cualquier caso, se ha creado esta clase, la cual es la encargada de crear, una vez asignado el movimiento y de forma dinámica, una zona o ícono en la esquina superior izquierda a través del cual se efectuará el movimiento. Este comportamiento es en todo momento configurable, por lo que su inclusión no es necesaria si no se desea e incluso se puede lograr que el elemento se pueda mover a través del ícono o de él mismo.

Las propiedades y métodos más importantes son:

- `crear`: Método que crea una zona o ícono dinámicamente, la cual posibilita el movimiento a través de ella y no del elemento.

- `eliminar`: Método que elimina la zona o ícono creado dinámicamente para posibilitar el movimiento a través de él.

- `tipo`: Propiedad que define si se va a utilizar o no el ícono para realizar el movimiento. Esta propiedad puede contener tres valores, 0 si se desea mover el elemento a través del ícono creado para esto, esta es la opción por defecto, 1 si se desea mover por el mismo elemento y 2 si desea ambos comportamientos.

Refactorización

- **`Elemento`**: En la colección de elementos guarda instancias de la clase `Fabrica` y no objetos que referencian al elemento HTML como lo hacía anteriormente.

3.2.4 Tercera iteración

Esta será la última iteración en el proceso de desarrollo del control.

- Convertir objeto ASP.NET a objeto JavaScript y viceversa.
- Persistir datos de estado.
- Identificar sesión de diseñador.
- Permitir la paginación de los datos.

3.2.4.1 Convertir objeto ASP.NET a objeto JavaScript y viceversa

Uno de los requisitos del control es que todos los cambios realizados tanto en el lado del servidor como en el lado del cliente sean persistidos para cada uno de los dos en sus diferentes medios. Para esto se implementa la clase `ApexSerializar` que utiliza entre sus métodos las funciones de .NET para la socialización de objetos ASP a JSON y viceversa. De esta manera el control al renderearse en el cliente se serializa el objeto JSON respectivo para cada ítem y cuando se hace una solicitud al servidor deserializa dicho JSON y se actualiza los datos para cada instancia del objeto ASP.NET.

Nuevas clases

- **ApexSerializar:** Se crea con el objetivo de lograr una fusión entre los objetos de JavaScript en notación JSON y los objetos de ASP.NET. De esta forma es posible intercambiar información entre el cliente y el servidor y viceversa.

Las propiedades y métodos más importantes son:

- `Prototipo`: Devuelve el prototipo del objeto a serializar.
- `GetObjFromJson`: Devuelve el objeto ASP.NET dado un objeto JavaScript en notación JSON.
- `GetJsonFromObj`: Devuelve un objeto JavaScript en notación JSON dado un objeto de la clase Prototipo.

Nuevos métodos

- **`Deserializar`:** Perteneciente a la clase `Contenedor`. Asigna los valores obtenidos del objeto cliente a este ítem.
- **`Serializar`:** Perteneciente a la clase `Contenedor`. Devuelve el objeto del ítem a serializar para enviar al cliente.

3.2.4.2 Persistir datos de estado

La implementación de esta funcionalidad permitirá salvar los cambios realizados en el componente una vez cerrada la página definiendo para ello una nueva fuente de datos llamada fuente de datos estado. Esto conlleva cambios en la arquitectura anterior para permitir que el componente Apex acepte dos fuentes de datos. Para cumplir los nuevos requerimientos se crean cuatro nuevas clases.

Nuevas clases

- **`ApexBaseDataBoundControl`:** Clase base para el control enlazado a datos que permite definir dos fuentes de datos e implementa las funcionalidades básicas para utilizarlas. Obliga a redefinir los siguientes métodos:

- `ActualizarEstado`: Define la lógica de como se actualizarán lo datos en la fuente de datos estado.

- `PerformSelect`: Define la lógica de acceso a datos para ambas fuentes de datos.

- `ValidateDataSource`: Valida una fuente de datos.

- `ApexDataBoundControl`: Clase que hereda de `ApexBaseDataBoundControl` implementando la lógica de acceso a datos y obligando a redefinir el método: `CreateChildControls`, esta nueva versión del método tendrá como parámetro las dos listas de datos recuperadas por el control durante el proceso de conexión a las fuentes de datos.

- `ApexBaseDataBoundControlDesigner`: Clase base para definir el comportamiento en tiempo de diseño de `ApexBaseDataBoundControl` declarando abstracto los métodos:

  - `ConnectToDataSource`: Define la lógica de conexión a las fuente de datos.

  - `CreateDataSource`: Define la lógica a implementar para crear una nueva fuente de datos.

  - `DataBind`: Establece como será el enlace a datos.

  - `DisconnectFromDataSource`: Desconecta el diseñador de datos actual.

  - `ApexDataBoundControlDesigner`: Hereda e implementa los métodos abstractos de la clase `ApexBaseDataBoundControlDesigner`. También define la interfaz de diseño para la asignación de las fuentes de datos y su vinculación con los controles definidos en cada una de las plantillas.

Nuevos métodos

- `ActualizarEstado`: Perteneciente a la clase `Apex`. Guarda las configuraciones que el usuario define como persistentes en la fuente de datos estado.

Refactorización

Como parte de los cambios realizados en la arquitectura se aplica un cambio en el esquema de solución propuesto anteriormente ya que ahora la clase `Apex` está extendiendo de `ApexDataBoundControl` y `ApexDisenador` de `ApexDataBoundControlDesigner`.

Identificar sesión de diseñador

El objetivo de esta historia de usuario es permitir identificar la sesión de diseño para usuarios diferentes dentro de la página y configurar la misma de acuerdo a cada una de sus preferencias.

Refactorización

- **`ActualizarEstado`:** Perteneciente a la clase `Apex`. Se modifica la manera de persistir los datos de estado para incluir un identificador de sesión que diferencia los ítems de igual nombre.

3.2.4.3 Permitir la paginación de los datos

Facilita el manejo de grandes cantidades de datos y la navegación entre estos. De esta forma es posible especificar la cantidad de elementos que se desea obtener en cada enlace con la base de datos. Permitiendo además hacer la paginación de los próximos o los anteriores elementos, pero siempre manteniendo la misma cantidad. Esto posibilita que cuando el resultado de una consulta a la base de datos devuelve muchos elementos, pues es posible ir viendo dichos resultados por partes.

Refactorización

- **`Apex`:** Se adiciona una nueva propiedad para definir la plantilla de paginación y se incorpora dentro de la lógica de creación de la jerarquía de controles en el método `CreateChlidControls`.

3.3 Conclusiones

A lo largo de esta fase se ha ido iterando sobre el control y agregándole nuevos comportamientos, Se han superado todas las pruebas hechas en cada una de las partes del proceso y se logra cumplir con el objetivo de lograr un producto final con un alto grado de aceptación y confiabilidad.

CAPÍTULO 4: APLICACIÓN DE LA FASE DE PRODUCCIÓN

4.1 Introducción

En esta fase el software pasa por pruebas adicionales y revisiones de rendimientos antes de ser trasladado al entorno del cliente. También se proponen nuevas ideas o sugerencias que son documentadas para su posterior implementación en una nueva versión o durante la fase de mantenimiento.

4.2 Fase de producción

Pruebas

Se realizan tres pruebas para evaluar el comportamiento del control en cada una de sus variantes y detectar cualquier posible error principalmente con el uso de las diferentes implementaciones de modelo DOM en cada uno de los navegadores.

- Representación y edición sincrónica de datos.
- Representación y edición Asincrónica de datos.
- Compatibilidad de modificaciones Cliente/Servidor.

4.2.1 Representación y edición sincrónica de datos

Caso 1: Fuente de datos 50 registros

Descripción

Para este caso de prueba se utilizan como gestores de bases de datos SQL, ACCESS para realizar una prueba de rendimiento con cincuenta registros. Se definen las plantillas de lectura, inserción, edición y paginado. Se configura el control para personalizar su comportamiento para cada uno de los valores posibles. En tiempo de ejecución se realizan las sobre el control las operaciones de insertar, editar, eliminar, paginar y mover. Para cada una de las configuraciones se evalúan el rendimiento, la persistencia y la configurabilidad gráfica del control.

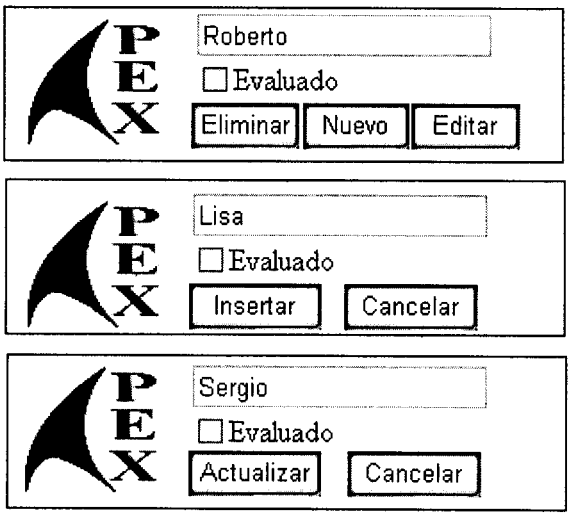


Figura 4.1 Interfaz del control para las pruebas.

Comportamiento

Para la primera prueba en la configuración no se incluye la paginación ni la persistencia de los datos en la fuente de datos estado. Los resultados obtenidos para los indicadores evaluados fueron:

Rendimiento: Se usó como herramienta para su evaluación Fiddler. El diseño del control contó con tres campos enlazados a datos, una imagen de 42 KB. Los resultados obtenidos como se puede ver a continuación resultaron satisfactorios.

Navegador	Tiempo menor	Tiempo mayor	Promedio
Interne Explorer 6	1.0781	2.0514	1.2656
Mozilla Firefox	0.7812	01.0156	0.9325

Tabla 4.1 Rendimiento en los navegadores con la cache deshabilitada.

Navegador	Tiempo menor	Tiempo mayor	Promedio
Interne Explorer 6	0.5187	0.6895	0.7343
Mozilla Firefox	0.5781	0.7832	0.5960

Tabla 4.2 Rendimiento en los navegadores con la cache habilitada.

Persistencia: Para obtener un resultado final de la evaluación de este parámetros se tomaron en cuenta diversos indicadores de acuerdo a las propiedades modificables en el control.

Indicador	Persistido	Descripción
DesbordeContenedor	Si	Se modificaron los valores Internet Explorer 6 presentó problemas en la implementación de su modelo DOM con el valor "auto".
IdContenedor	Si	No se registraron problemas.
EstiloItem	No	No persistieron los cambios en el estilo de línea.
MostrarHolder	Si	No se registraron problemas.
TipoHolder	Si	No se registraron problemas.
OpacidadHolder	Si	No se registraron problemas.
UrlImagenHolder	Si	No se registraron problemas.
TipoMovimiento	Si	No se registraron problemas.

Tabla 4.3 Prueba de persistencia 1.

Solo se detectaron dos errores durante las pruebas, que fueron solucionados como parte de las mismas.

Configurabilidad: El control fue presentado al cliente y aceptado todo el proceso de su gestión en tiempo de diseño por el programador.

En la segunda prueba realizada dentro de este caso de pruebas se incluyó dentro de la configuración del control las opciones de paginado persistencia de datos en una fuente de datos estados definida por el usuario. Se utilizaron como gestores de fuentes de datos los mismos del caso anterior. Los resultados obtenidos fueron:

Rendimiento: El control presentó rendimientos similares durante el proceso de carga de la página para los distintos navegadores.

Persistencia: La prueba de persistencia de la configuración del control se aplicó sobre las propiedades mostradas a continuación.

Propiedad	Persistido	Descripción
IndicePagina	Si	No se presentaron errores.
Paginar	Si	No se presentaron errores.
PosicionPaginado	Si	No se presentaron errores.
TamañoPagina	Si	No se presentaron errores.
PersistirEstado	Si	Al no tener definida una fuente de datos de estado solo persisten los estados mientras estén en la misma página.

Tabla 4.4 Prueba de persistencia 2.

Configurabilidad: Todas las opciones de configuración en tiempo de diseño para las propiedades adicionadas en esta prueba han sido aceptadas por el cliente.

Caso 2: Fuente de datos vacía

Descripción

En el segundo caso de esta prueba se define una fuente de datos vacía que puede ser como resultado del inicio de gestión de la aplicación o por eliminación de todos los datos representados. Solo se evaluaron en esta prueba el comportamiento con la implementación de la plantilla utilizada para estos casos.

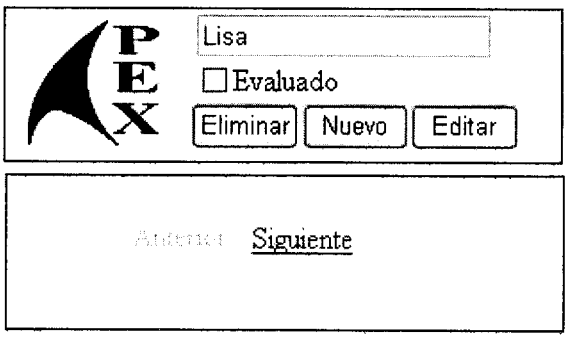


Figura 4.2 Interfaz del control con paginación.

Comportamiento

En este caso de prueba se detecta un error lógico generado al tratar de entrar en modo de diseño desde la plantilla vacía y es corregido durante el proceso.

4.2.2 Representación y edición Asincrónica de datos

Descripción

Para la el trabajo asincrónico el control es utilizado dentro de un UpdatePanel. Se mantiene el diseño gráfico y la implementación de las plantillas de la prueba anterior y se hacen varias configuraciones de las propiedades que definen el comportamiento del control.

Comportamiento

No se presentaron errores en las operaciones de edición de los datos, ni en la persistencia de las propiedades gráficas. Los tiempos de carga de la página se mantuvieron en el mismo rango que los obtenidos de manera sincrónica para el proceso de carga y disminuyeron dentro de los eventos de respuesta a solicitudes del control.

4.3 ¿Como usar Apex?

Una vez terminada las pruebas sobre el control éste ya está listo para pasar al entorno del cliente. En este epígrafe se explicará cómo usar Apex, los requisitos para su funcionamiento y las prestaciones brindadas.

4.3.1 Requisitos

El control Apex fue creado para la versión 2.0 de ASP.NET, necesita la librería ASP.NET AJAX la cual es una librería libre brindada por Microsoft de manera independiente para la versión 2.0 de ASP.NET y puede descargarse en la dirección http://www.asp.net.

Pasos a seguir:

1. Descargar la librería.
2. Ejecutar instalador.
3. Comprobar la inclusión de los nuevos controles en la barra de herramientas de Visual Studio.NET 2005.

4.3.2 Instalación del control Apex

A continuación se describen los pasos a seguir para la inclusión del control Apex dentro del entorno de desarrollo de Visual Studio.NET.

1. Abrir la barra de herramientas de Visual Studio.NET.
2. Hacer click derecho, y elegir Shoose Item.
3. En el formulario mostrado seleccionar la pestaña .NET Framework Components.
4. Hacer click en el botón Browse para seleccionar la dirección de origen de la dll del control y aceptar.
5. Arrastrar el control al área de diseño para su utilización.

4.3.3 Definir una instancia del control

Existen tres vías para la inclusión de una instancia del control Apex dentro de una página web.

Primera:

Arrastrar el control desde la barra de controles hasta la región de diseño o hacer doble click sobre el ícono del control en la caja de herramientas o toolbox.

Segunda:

Primeramente se incluye la referencia del proyecto a la dll del control. Luego se declara en la cabecera de la página en la vista de código una referencia al ensamblado. Por ultimo se crea el control mediante la definición de las etiquetas HTML correspondientes.

Tercera:

Se incluye la referencia del proyecto a la dll del control. Luego se declara en la cabecera de la página en la vista de código una referencia al ensamblado. Se implementan los métodos necesarios en el servidor para crear el control dinámicamente y agregarlo a la página en tiempo de ejecución.

4.3.4 Definir Fuentes de Datos

El control Apex acepta como fuentes de datos válidas las definidas dentro del marco de trabajo .NET como controles de orígenes de datos.

- XmlDataSource.
- ObjectDataSource.
- SqlDataSource.
- AccesDataSource.

Fuente de datos vista:

Pasos a seguir para definir la fuente de datos a mostrar por el control

1. Arrastrar el control de origen de datos a la página web.
2. Configurar la conexión a la fuente de datos.
3. Establecer la consulta para seleccionar los datos.
4. Para permitir eliminar, insertar y actualizar hacer click en el botón Advance y marcar la opción en las fuentes de datos que lo soporten.
5. Aceptar y salir. (Ver Anexo 6).

Fuente de datos estado:

Pasos a seguir para definir la fuente de datos para salvar el estado del control.

1. Arrastrar el control de origen de datos a la página web.

2. Configurar la conexión a la fuente de datos.

3. Establecer la consulta para seleccionar los datos. Existen dos escenarios posibles para la fuente de datos estado.

   - **No existe sesión de diseño:** Se establece la consulta para la selección de los datos. Para permitir eliminar, insertar y actualizar hacer click en el botón Advance y marcar la opción. Esta opción es un requisito para el correcto funcionamiento de la fuente de datos estado.

   - **Existe sesión de diseño:** Se establece la consulta para la selección de los datos. Cuando se va a utiliza una sesión de diseño se ha de especificar en la cláusula `where` de la selección, el campo que contiene el identificador de la sesión. La tabla donde se guarden los datos de estado debe estar identificada por dos campos uno para el identificador del ítem y otro para el de la sesión.

4 Para permitir eliminar insertar y actualizar hacer click en el botón Advance y marcar la opción. Verificar en la consulta de eliminar que no requiera el parámetro para el campo que guarda los valores que identifican la sesión debido a que esta operación se realizará para todas las sesiones guardadas tan solo por el identificador del ítem.

5 Aceptar y salir.

Para asignar las fuentes de datos definidas al control:

Abrir el panel de etiquetas inteligentes y seleccionar sus identificadores en las listas desplegables mostradas o abrir la barra de propiedades y seleccionar los identificadores para las propiedades correspondientes (`FuenteDatosVista`, `FuenteDatosEstado`). (Ver Figura 4.3 y 4.4).

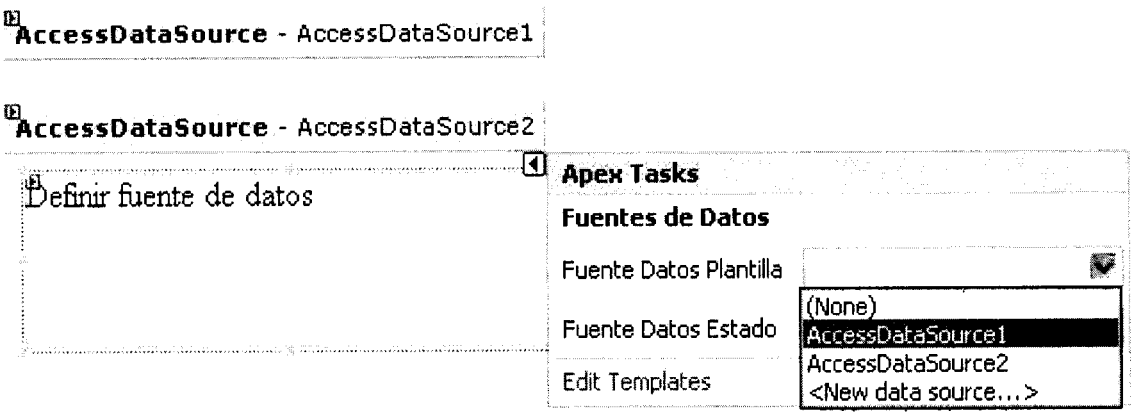


Figura 4.3 Definir fuentes de datos.

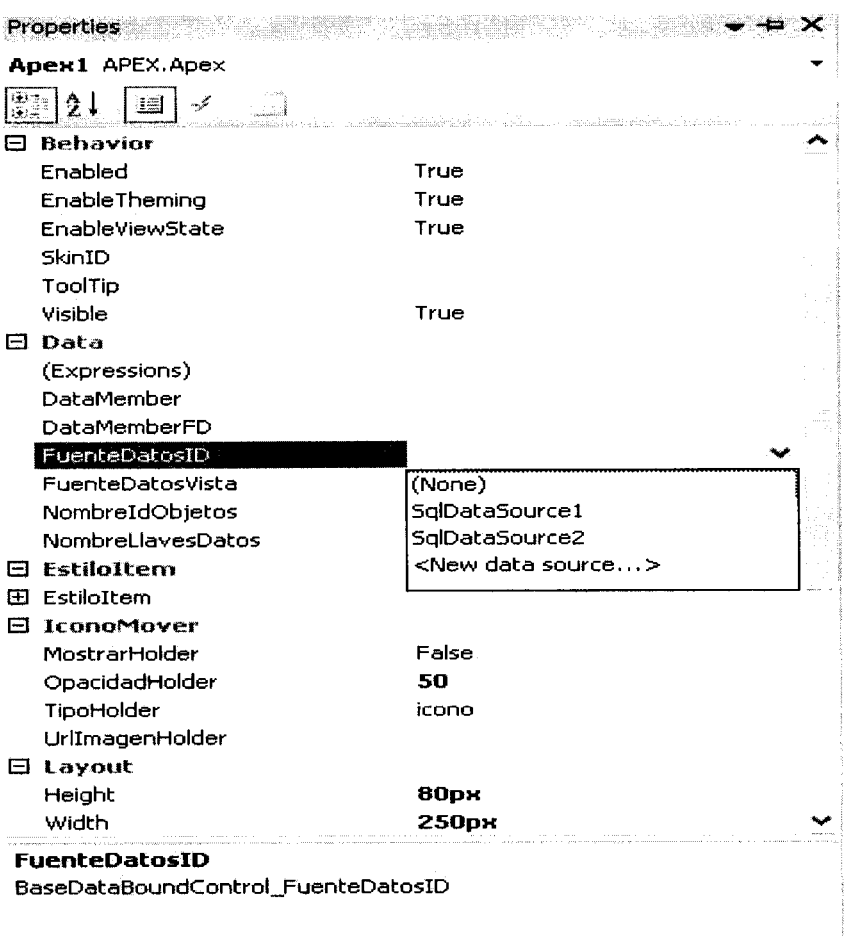


Figura 4.4 Definir Fuente de datos, barra de propiedades.

4.3.5 Diseño

La configuración gráfica del control se logra mediante la definición de plantillas. El control cuenta con 5 plantillas y el programador debe definir la interfaz para cada una de ellas en función de las funcionalidades requeridas, solamente la plantilla de lectura es necesaria.

Para la edición de las plantillas abrir el panel de diseño elegir `Edit Template` y seleccionar la plantilla a editar. Las plantillas permiten tanto controles del servidor como controles HTML pero solo los primeros son compatibles con enlace a datos.

Durante la edición de las plantillas debe tenerse en cuenta el comportamiento de los controles en función de la posición absoluta o relativa. Se recomienda el uso de un panel como contenedor en posición absoluta para facilitar la representación de varios controles dentro de la plantilla.

La plantilla `Lectura`: Es donde se muestran los datos por defecto, constituye un requisito para el correcto funcionamiento del control. Dentro de ella es posible definir controles de tipo Button, LinkButton o ImageButton con sus propiedades `CommandArgument` en: `Nuevo`, `Eliminar`, `Seleccionar` y `Editar` para moverse dentro de los diferentes modos que permite el control Apex. Esta plantilla no admite controles con la propiedad `CommandArgument` en uno de los siguientes valores: `Actualizar`, `Cancelar`, `Insertar`, `Siguiente`, `Anterior`.

La plantilla `Editar`: Plantilla que se va a utilizar para la edición de los datos, para su completo funcionamiento se deben insertar dos controles de tipo Button, LinkButton o ImageButton con el valor de su propiedad `CommandArgument` en: `Actualizar` para realizar la operación sobre la fuente de datos y `Cancelar` para volver al modo de lectura. Esta plantilla no admite controles con la propiedad `CommandArgument` en uno de los siguientes valores: `Nuevo`, `Editar` `Insertar`, `Siguiente`, `Anterior`.

La plantilla `Insertar`: Plantilla que se va a utilizar para la inserción de nuevos datos, para su completo funcionamiento se deben insertar dos controles de tipo Button, LinkButton o ImageButton con el valor de su propiedad `CommandArgument` en: `Insertar` para realizar la operación sobre la fuente de datos y `Cancelar` para volver al modo de lectura. Esta plantilla no admite controles con la propiedad `CommandArgument` en uno de los siguientes valores: `Nuevo`, `Editar`, `Actualizar`, `Siguiente`, `Anterior`.

La plantilla `Vacia:` Se muestra cuando no existen registros en la fuente de datos. En ella se debe definir un control de tipo Button, LinkButton o ImageButton con el valor de su propiedad `CommandArgument` en `Nuevo` para cambiar a modo de inserción. Esta plantilla no admite controles con la propiedad `CommandArgument` en uno de los siguientes valores: `Editar, Eliminar, Actualizar, Cancelar.`

La plantilla paginar: Define la interfaz para la paginación del control y para ello debe tener un control con el valor de su propiedad `CommandArgument` en `Siguiente` y otro con su valor de propiedad `CommandArgument` en `Anterior.` Esta plantilla no admite controles con la propiedad `CommandArgument` en uno de los siguientes valores: `Nuevo, Editar, Eliminar, Actualizar, Cancelar, Insertar.`

Al terminar el proceso de creación de las plantillas ir al panel de diseño y seleccionar `End Template Editing.`

Nota: Para enlazar los controles a los campos de la fuente de datos seguir la lógica que brinda el marco de trabajo .NET para los controles enlazados a datos. (Ver Anexo 6).

4.3.6 Personalización

La personalización del comportamiento del control en tiempo de diseño se logra mediante la configuración de las propiedades mostrada en el panel de propiedades. El funcionamiento de cada una de ellas se encuentra detallado en la tabla. (Ver Anexo 7).

4.3.7 Ejecución

Durante la ejecución de la página el usuario final podrá hacer uso de todas las funcionalidades del control de acuerdo al valor de configuración dado por el desarrollador. Los desarrolladores pueden en tiempo de ejecución modificar el comportamiento de Apex accediendo al objeto del control tanto en el servidor como en el cliente.

Para acceder o modificar las propiedades del comportamiento para cada ítem en el servidor representado por Apex se utiliza la sintaxis:

```
NombreControl.Controls[posición].Propiedad;
```

Ejemplo

```
Apex1.Controls[2].TipoHolder = EtipoHolder.icono;
```

Debe destacar que los valores de la mayoría de las propiedades disponibles en el servidor son de tipo enumerador. (Ver Anexos 7 y 8).

También se puede controlar el ciclo de ejecución del control en el servidor mediante los eventos que son lanzados después de realizar alguna acción de interés para el programador. (Ver Tabla 3.1 Controladores de eventos en Apex).

Para modificar el comportamiento de cada elemento en el cliente mediante JavaScript se accede a las propiedades y métodos de cada uno mediante el objeto Apex definido cuyo nombre será igual al valor del identificador especificado para el control.

Para acceder a un ítem contenido en Apex

Existen tres formas para acceder a un ítem definido en Apex mediante JavaScript

1. `NombreControl.Elemento.coleccion['idElemento'];`
2. `NombreControl.Elemento.coleccion.idElemento;`
3. `NombreControl.Elemento.obtener(0);`

Movimiento

Para acceder al elemento se pueden utilizar cualquiera de las sintaxis definidas anteriormente.

- **Mover un elemento**.

  ```
  NombreControl.Elemento.coleccion['idElemento'].mover();
  ```

  Ejemplo:

  ```
  Apex1.Elemento.coleccion['personal'].mover ();
  ```

- **Detener un elemento.**

  ```
  NombreControl.Elemento.coleccion['idElemento'].detener();
  ```

  Ejemplo:

  ```
  Apex1.Elemento.coleccion['personal'].detener();
  ```

- **Detener todos los elementos al mismo tiempo.**

```
NombreControl.Elemento.detenerTodo();
```

Ejemplo:

```
Apex1.Elemento.detenerTodo();
```

- **Eliminar un elemento de la colección de elementos.**

```
NombreControl.Elemento.eliminar('idElemento');
```

Ejemplo:

```
Apex1.Elemento.eliminar('personal');
```

- **Eliminar todos los elementos de la lista.**

```
NombreControl.Elemento.eliminarTodo();
```

Ejemplo:

```
Apex1.Elemento.eliminarTodo();
```

- **Cambiar el tipo de movimiento.**

```
NombreControl.Elemento.coleccion['idElemento'].item.tipoMovimi
ento =      tipoMov;
```

Nota: tipoMov deber tener un valor del enumerado `$EtipoMov`. Anexo 9.

Ejemplo:

```
Apex1.Elemento.coleccion['personal'].item.tipoMovimiento
  =$ETipoMov.total;
Apex1.Elemento.coleccion['personal'].item.tipoMovimiento     =
'TTL';
```

Holder

- **Cambiar el tipo de Holder.**

```
NombreControl.Elemento.coleccion['idElemento'].holder.tipo    =
tipoHolder;
```

Nota: tipoHolder deber tener un valor del enumerado `$EtipoHolder`. Anexo 9.

Ejemplo:

```
Apex1.Elemento.coleccion['persona1'].holder.tipo =
$ETipoHolder.ambos;
```

- **Opacidad del Holder.**

```
NombreControl.Elemento.coleccion['idElemento'].holder.opacidad
= (0 - 100);
```

Ejemplo:

```
Apex1.Elemento.coleccion['persona1'].holder.opacidad = 60;
```

- **Holder siempre visible.**

```
NombreControl.Elemento.coleccion['idElemento'].holder.mostrar
= true;
```

Ejemplo:

```
Apex1.Elemento.coleccion['persona1'].holder.mostrar  = true;
```

- **Cambiar la imagen por defecto del Holder.**

```
NombreControl.Elemento.coleccion['idElemento'].holder.imagen
='UR';
```

Ejemplo:

```
Apex1.Elemento.coleccion['persona1'].holder.imagen
='/desconocido.jpg';
```

Actualizar

- **Actualizar elemento.**

```
NombreControl.Elemento.coleccion['idElemento'].actualizar();
```

Ejemplo:

```
Apex1.Elemento.coleccion['persona1'].actualizar();
```

- **Actualizar todos los elementos.**

```
NombreControl.Elemento.actualizarTodo ();
```

Ejemplo:

```
Apex1.Elemento.actualizarTodo();
```

Relaciones

- La clase Apex cuenta con los métodos necesarios para representar una relación entre dos elementos, en el anexo 10 se especifica un ejemplo. Para accederlos se utiliza la sintaxis:

```
NombreControl.Relacion.Metodo ();
```

Ejemplo:

```
Apex1.Relacion.anchoLinea = 2;
```

Prototipo

- En esta clase se encuentran todas las propiedades a persistir cuando se realiza una solicitud al servidor. Se pueden acceder a ellas directamente mediante la sintaxis:

```
NombreControl.Elemento.coleccion['idElemento'].Prototipo.
['nombre de la propiedad'];
```

Además contiene la colección `Items` que guarda todos los datos extra que desee el usuario.

- Guardar datos extras en `Prototipo` para la persistencia.

```
NombreControl.Elemento.coleccion['idElemento'].Prototipo.agreg
ar (key, value);
```

Ejemplo:

```
Apex1.Elemento.coleccion['personal'].Prototipo.agregar('nombre
', 'José');
```

`key`: Nombre que será el identificador del elemento en la colección Ítems de la clase Prototipo.

`value`: Valor que se desee persistir.

Para persistir este dato en la fuente de datos encargada de la persistencia, es necesario nombrar un campo de la misma con este nombre y posteriormente hacer un enlace a datos a través de la interfaz visual del control Apex diseñada para esta funcionalidad.

Para especificar un contenedor o área de trabajo

Es necesario especificar el identificador de un panel o un `div` para que todos los resultados de la consulta que realice el control Apex sean mostrados dentro de la región.

- **Agrupar un elemento.**

```
NombreControl.Elemento.coleccion['idElemento'].
asignarGrupo(idContenedor, conf);
```

Ejemplo:

```
Apex1.Elemento.coleccion['personal'].
asignarGrupo("div1","auto");
```

`idContenedor`: Identificador del panel o `div` en que se desea que estén enmarcados los ítem representados.

`conf`: Comportamiento que se desea una vez que la cantidad de elementos exceda los límites de la región comprendida por el panel o el `div` o cuando a un elemento se le aplica movimiento y también sobrepase estos límites. Su valor por defecto es '`auto`'. (Anexo 9).

Eventos

- Todos los eventos tienen la estructura: `on + [nombre del evento].`
- Todos los métodos encargados de agregar manejadores definidos por el usuario a dichos métodos tienen la estructura: `add + [nombre del evento] + Handler.`
- Todos los métodos encargados de eliminar un manejador de un evento en específico tiene la estructura: `remove + [nombre del evento] + Handler.`

Ver anexo 11.

Nota: El nombre de cada evento se encuentra en la tabla de eventos correspondientes de la clase `Elemento` (Anexo 4) `y Fabrica` (Anexo 3).

4.4 Conclusiones

En este capítulo se ha hecho referencia a la aplicación de la fase de producción de XP. En dicha fase se han aplicado algunos casos de pruebas, las cuales han permitido obtener una estadística del comportamiento del control durante su transcurso, ya sea en las diferentes pruebas de persistencia o como en las de rendimiento observadas en los distintos navegadores.

CONCLUSIONES

Con la culminación de este trabajo queda finalizada la primera versión del control Apex que sirve de marco de trabajo para la representación y edición gráfica de datos en páginas Web ASP.NET. Entre los logros más importantes alcanzados están:

- Integrar en el control el trabajo tanto en el cliente como en el servidor.
- Permitir el intercambio sincrónico y asincrónico de la información.
- Independizar el diseño gráfico para ampliar la capacidad de personalización del control.
- El manejo del ciclo de vida del control mediante la ejecución de eventos.
- La persistencia de la configuración gráfica.
- La gestión de los datos mostrados (eliminar, actualizar, insertar).
- La capacidad de incorporar nuevas funciones ejemplo: Las relaciones.
- Configurabilidad total en tiempo de diseño.
- La utilización de objetos script en el cliente.
- La persistencia de los cambios realizados en el cliente en el servidor y viceversa.

Por todas estas características el control permite reducir el tiempo de desarrollo, brindando una lógica de trabajo intuitiva, reutilizable y fácil de extender por los desarrolladores que pueden contar con un marco de trabajo flexible que integra las dos grandes ramas del desarrollo Web, cliente y servidor.

RECOMENDACIONES

Una vez concluido el desarrollo de este trabajo investigativo se recomienda:

- Desarrollar las funcionalidades necesarias para independizar el trabajo asincrónico del control.
- Implementar nuevas funciones sobre el marco de trabajo creado por el control para aumentar sus funcionalidades o posibilidades. Ejemplo: Permitir mover el elemento solamente hacia un contenedor definido.
- Implementar una nueva clase base que permita la representación de varias fuentes de datos en un control.
- Definir plantillas predefinidas para cada funcionalidad relacionada con ellas.
- Ampliar las posibilidades de fuentes de datos de estado para permitir la persistencia en XML.

BIBLIOGRAFÍA CITADA

1. **Camallea, Noel Luis Núñez.** *Programar para Internet con ASP.NET.* Ciudad de La Habana : Científico - Técnica, 2004. ISBN-959-05-0363-2.

2. **World Wide Web Consortium Oficina Española.** Guía Breve de CSS. [En línea] 09 de enero de 2008. [Citado el: 07 de febrero de 2008.] http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo.

3. —. Guía Breve de Tecnologías XML. [En línea] 09 de enero de 2008. [Citado el: 04 de febrero de 2008.] http://www.w3c.es/divulgacion/guiasbreves/TecnologiasXML.

4. **Brun, Ricardo Eíto.** Hipertext.net. [En línea] mayo de 2005. [Citado el: 03 de marzo de 2008.] http://www.hipertext.net/web/pag256.htm#Una%20breve%20introducci%F3n%20al%20lenguaje%20XML.

5. JSON. [En línea] 02 de mayo de 2008. [Citado el: 12 de marzo de 2008.] http://www.json.org/json-es.html.

6. **World Wide Web.** Especificación del Núcleo del Modelo de Objetos del Documento (DOM). [En línea] 24 de marzo de 2005. [Citado el: 13 de febrero de 2008.]

7. **Mora, Sergio Lujan.** *Programación de aplicaciones web Historia, principios básicos y clientes web.* 2002. 9788484542063 .

8. Manifesto for Agile Software Development. [En línea] 27 de agosto de 2007. [Citado el: 23 de febrero de 2008.] http://www.agilemanifesto.org/.

9. **Martin, Robert y Newkirk, James.** *La Programación Extrema en la Práctica.* s.l. : Addison Wesley, 2002.

10. MSDN Información general sobre los atributos. [En línea] 15 de enero de 2008. [Citado el: 23 de marzo de 2008.] http://msdn.microsoft.com/es-es/library/cc467867(VS.71).aspx.

11. MSDN Información general sobre los diseñadores de controles ASP.NET. [En línea] 21 de febrero de 2008. [Citado el: 21 de febrero de 2008.] http://msdn.microsoft.com/es-es/library/wxh45wzs.aspx.

12. MSDN Introducción a los controles Web personalizados. [En línea] 15 de enero de 2008. [Citado el: 20 de febrero de 2008.] http://msdn.microsoft.com/es-es/library/cc437541(VS.71).aspx.

BIBLIOGRAFÍA CONSULTADA

B. McClure, Wallace, y otros. 2007. Beginning ASP.NET 2.0 AJAX. Indianapolis : Wiley Publishing,, 2007. 978-0-470-11283-0.

Gibbsand, Matt y Wahlin, Dan. 2007. Professional ASP.NET 2.0 AJAX. Indianapolis : Wiley Publishing, 2007. 9780470109625.

Goodman, Danny. 2001. JavaScript Bible. New York : Hungry Minds, 2001. 0-7645-4718-6.

Khosravi, Shahram. 2007. ASP.NET AJAX Programmer's Reference with ASP.NET 2.0 or ASP.NET 3.5. Indianapolis : Wiley Publishing, 2007. 978-0-470-10998-4.

MacDonald, Matthew. 2005. Beginning ASP.NET 2.0 in C# 2005 From Novice to Professional. s.l. : Apress, 2005. 1-59059-572-6.

MSDN. 2008. Editores de tipos de interfaz de usuario. [En línea] 07 de febreo de 2008. [Citado el: 06 de abril de 2008.] http://msdn.microsoft.com/es-es/library/ms171838.aspx.

—. 2008. Ejemplo de convertidor de tipos. [En línea] 07 de febrero de 2008. [Citado el: 06 de abril de 2008.] http://msdn.microsoft.com/es-es/library/87926x56(VS.80).aspx.

Roodyn, Neil. 2004. eXtreme .NET: Introducing eXtreme Programming Techniques to .NET Developers. s.l. : Addison Wesley Professional, 2004. 0-321-30363-6.

Vogel, Peter. 2006. Web Parts and Custom Controls. Indianapolis : Wiley Publishing, 2006. 978-0-7645-7860-1.

Welicki, León. 2003. Implementando Extreme Programming en la Plataforma .NET. 2003. 1501032003-3819.

GLOSARIO DE TÉRMINOS

Capability Maturity Model (CMM): Modelo de calidad enfocado a la mejora de procesos en un proyecto, división u organización que se basa en tener modelos de referencia basados en prácticas maduras. Este modelo es definido por el SEI (Software Engineering Institute) de la Universidad Carnegie Mellon.

Cookies: Son pequeños archivos de datos que un sitio Web guarda en el cliente para salvar la información de estado que se ha producido.

CommandArgument: Propiedad que obtiene o establece un argumento opcional que se pasa al controlador de eventos.

Common Language Runtime (CLR): Es el motor en tiempo de ejecución del .NET Framework.

Common Gateway Interface (CGI): Estándar que permite el intercambio de información entre un servidor y un programa externo al servidor. Un programa CGI es un programa preparado para enviar datos desde y hacia un servidor web según este estándar.

Controles de origen de datos: Son controles de servidor presentados por Microsoft Visual Studio 2005 que definen la lógica de acceso a datos.

Controles enlazados a datos: Son controles que pueden ser enlazados a un control de origen de datos y utilizarlos para definir su interfaz.

Deserializar: Proceso inverso de serializar.

DesignerVerb: Un verbo de diseñador es un comando de menú vinculado a un controlador de eventos que se agregan al menú contextual de un componente en tiempo de diseño.

Diseñadores: Clases que proporcionan lógica que puede ajustar la apariencia o comportamiento de un tipo en tiempo de diseño en ASP.NET.

DIV: Etiqueta para definir un elemento HTML que fue introducida en la especificación de HTML 4 para proveer un mecanismo genérico para agrupar y dar estructura a los documentos.

Drag&Drop: Acción de arrastrar y soltar. Recurso que permite que los usuarios muevan objetos desde una parte de la página Web hacia otra.

ECMA – 262: Es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por Netscape Communications Corporation.

Global.asax: Archivo opcional que contiene código para responder a eventos del nivel de la aplicación y de la sesión provocados por ASP.NET o por módulos HTTP.

HttpApplication: Clase que define los métodos, las propiedades y los eventos comunes a todos los objetos de aplicación incluidos en una aplicación ASP.NET.

HttpContext: Clase que encapsula toda la información específica de HTTP acerca de una solicitud HTTP individual en ASP.NET.

HttpRequest: Clase que permite a ASP.NET leer los valores HTTP enviados por un cliente durante una solicitud Web.

HttpResponse: Clase que contiene la información de la respuesta HTTP de una operación ASP.NET.

ISAPI: Interfaz de programación de aplicaciones para servidores de Internet. Un conjunto de funciones para servidores de Internet, como Microsoft Windows NT Server ejecutando Microsoft Internet Information Server (MIIS).

Lenguajes script: Lenguajes de programación diseñados para ser ejecutado por medio de un intérprete.

Placeholder: Control de ASP.NET que actúa como un contenedor para almacenar en la página Web los controles de servidor agregados dinámicamente.

Plantillas: Una plantilla es un conjunto de elementos y controles HTML que componen el diseño de una parte determinada de un control. ASP.NET brinda soporte a estas características mediante la definición de propiedades de tipo ITemplate.

Renderear: Adaptación al castellano del vocablo inglés rendering y que define un proceso en la programación web mediante el cual el servidor transforma la página a lenguaje HTML válido para mostrarse en el cliente.

Rollovers: Efecto de cambiar una imagen por otra al pasar el cursor sobre ella. Generalmente se utiliza en botones y otros elementos gráficos para mostrar que son hipervínculos.

ScriptManager: Control que administra las bibliotecas de scripts y los archivos de script AJAX de ASP.NET, la representación parcial de página y la generación de la clase de proxy de cliente para los servicios Web y de aplicación.

Serializar: Permite al desarrollador guardar el estado de un objeto y volver a crearlo cuando es necesario. Proporciona almacenamiento de objetos e intercambio de datos.

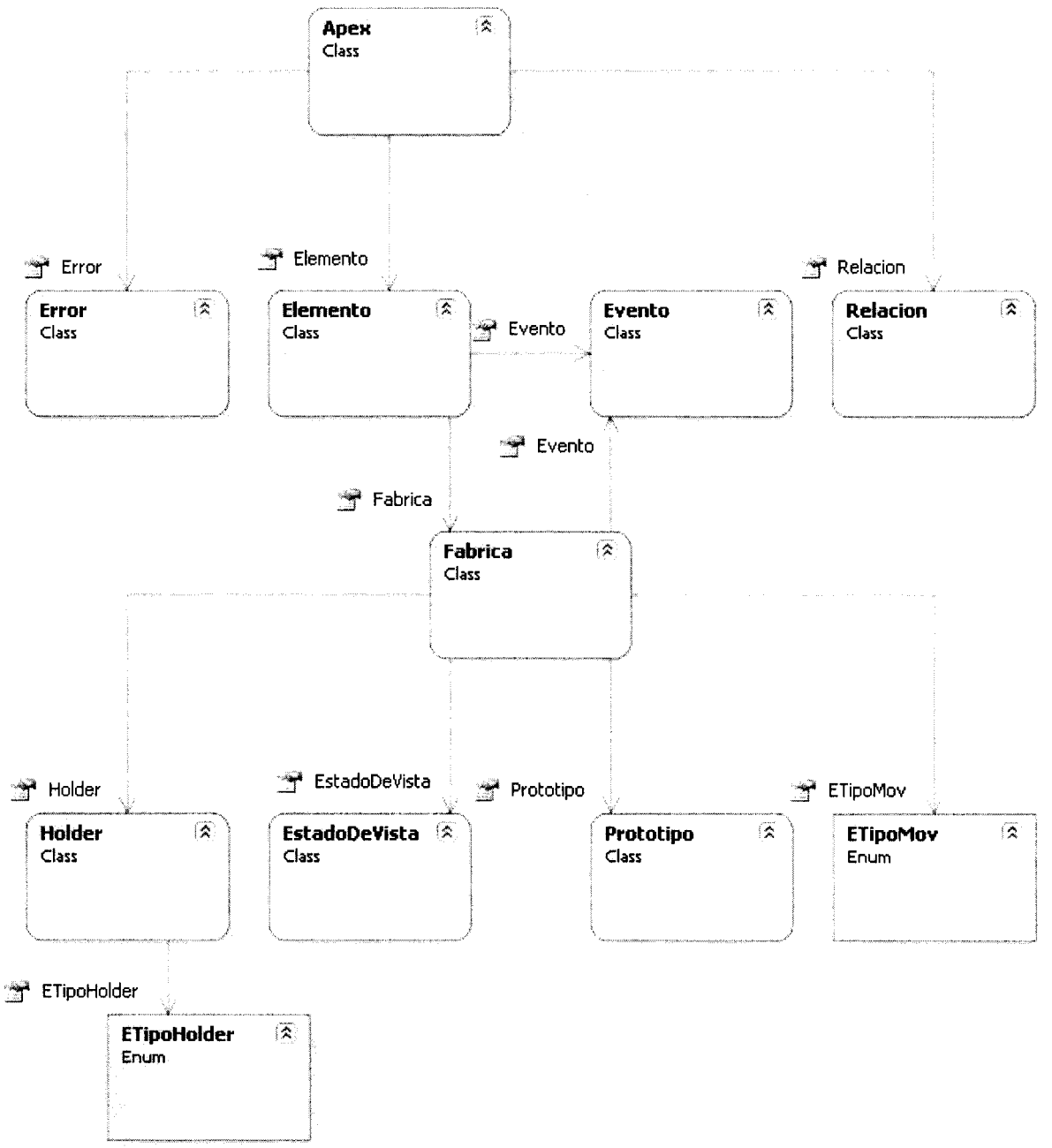
Sun Microsystems: Empresa ubicada en Mountain View, California que fabrica hardware y software para computadoras. Es mejor conocida en los últimos años por desarrollar el lenguaje de programación, Java.

UpdatePanel: Control de ASP.NET con funcionalidades AJAX usado para habilitar la representación parcial de una página.

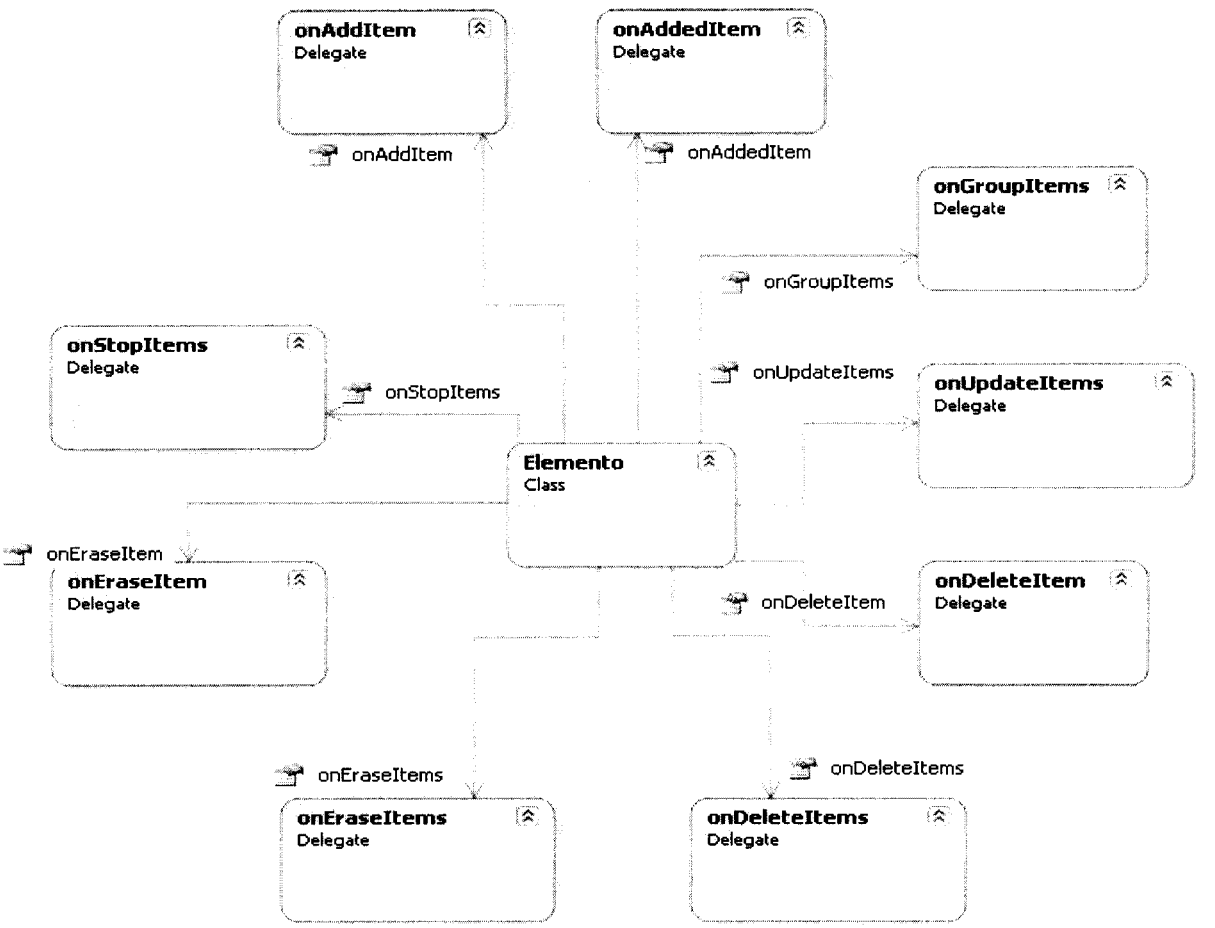
WYSIWYG: Acrónimo en ingles de What You See Is What You Get, lo que ves es lo que hay. Los programas WYSIWYG, son aquellos que permiten manipular el trabajo y además poder ir viéndolo de la forma en que quedará finalmente.

ANEXOS

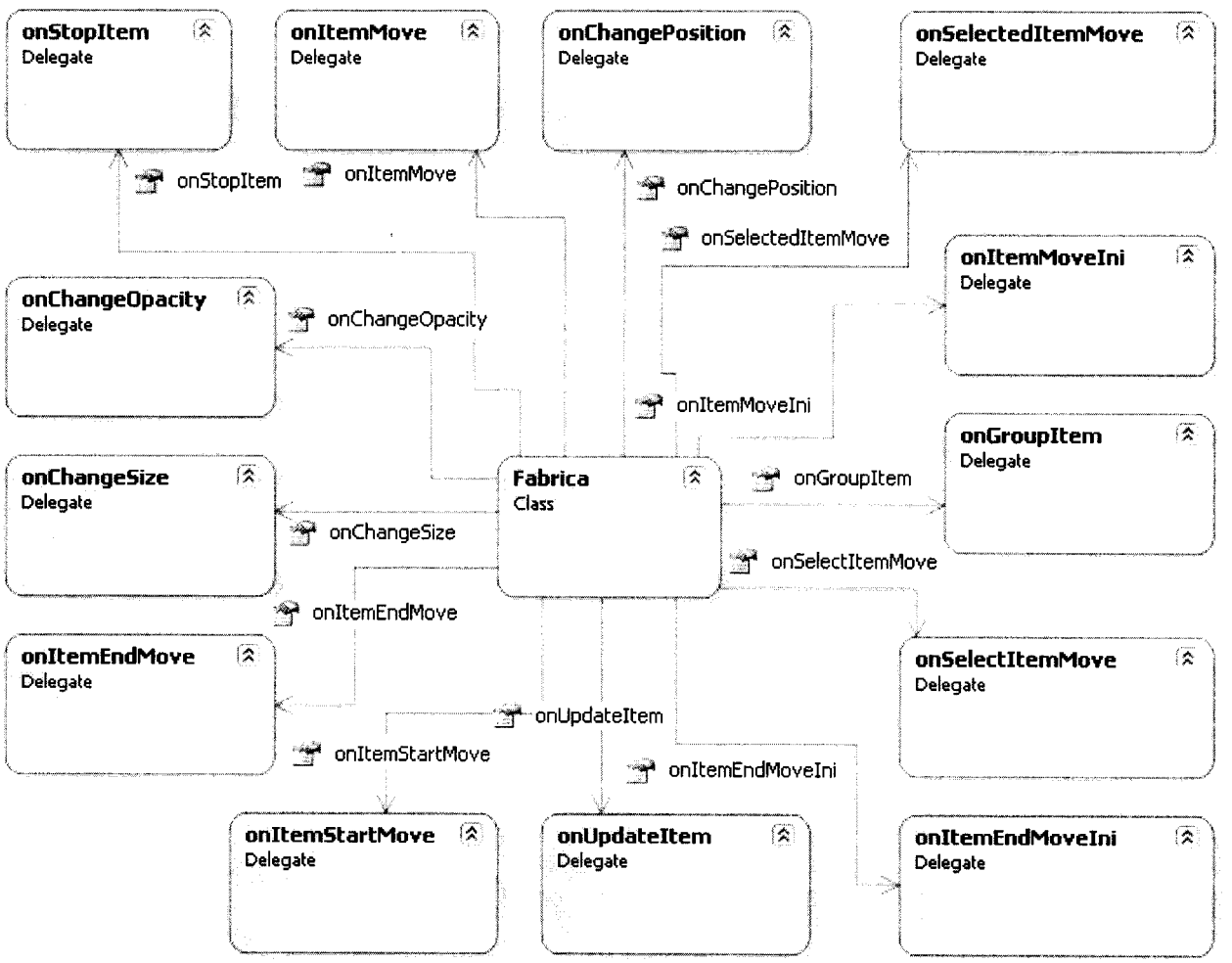
Anexo 1: Diagrama de clases de la librería JavaScript del control Apex.



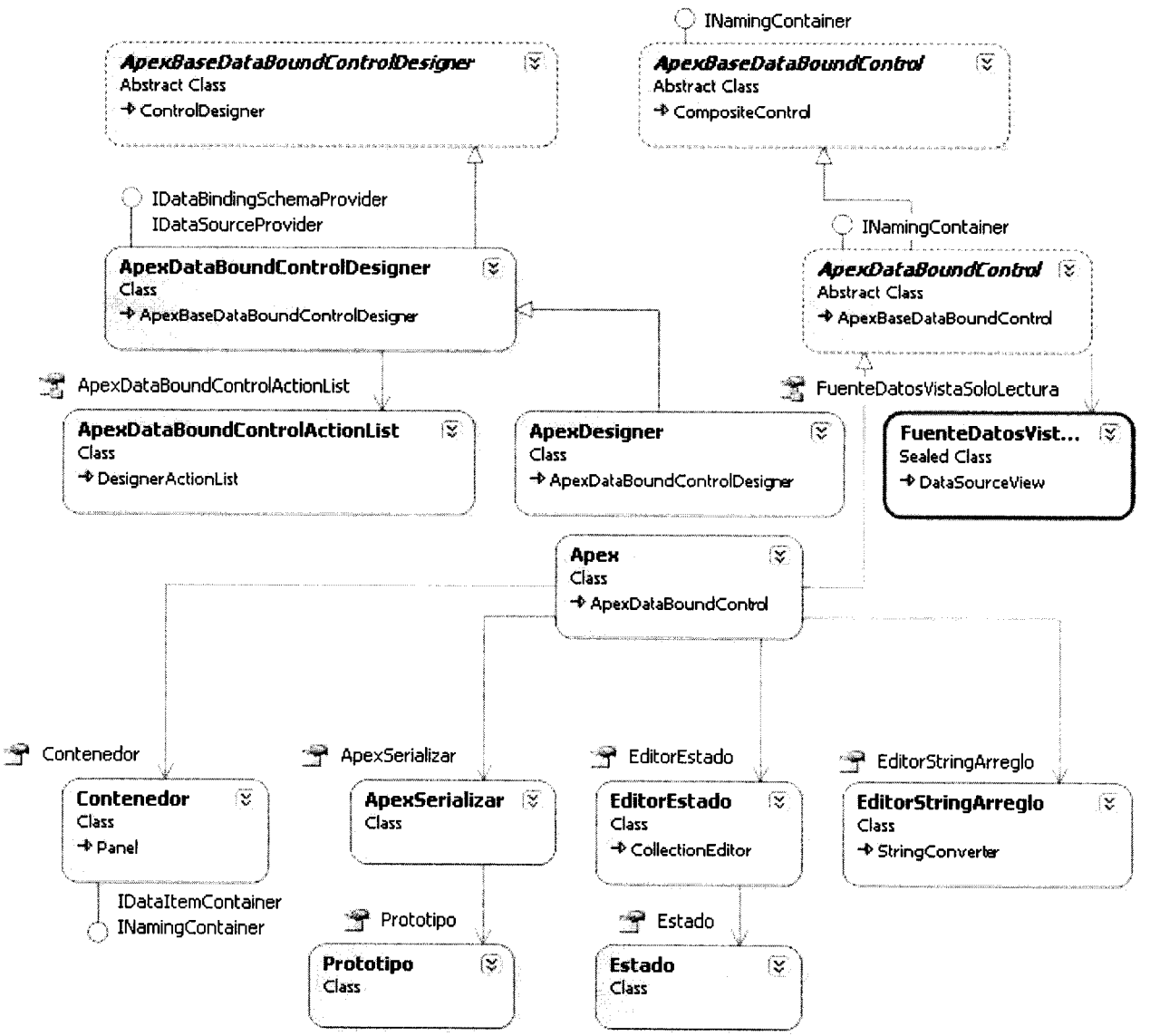
Anexo 2: Diagrama de eventos de la clase Elemento de la librería JavaScript del control Apex.



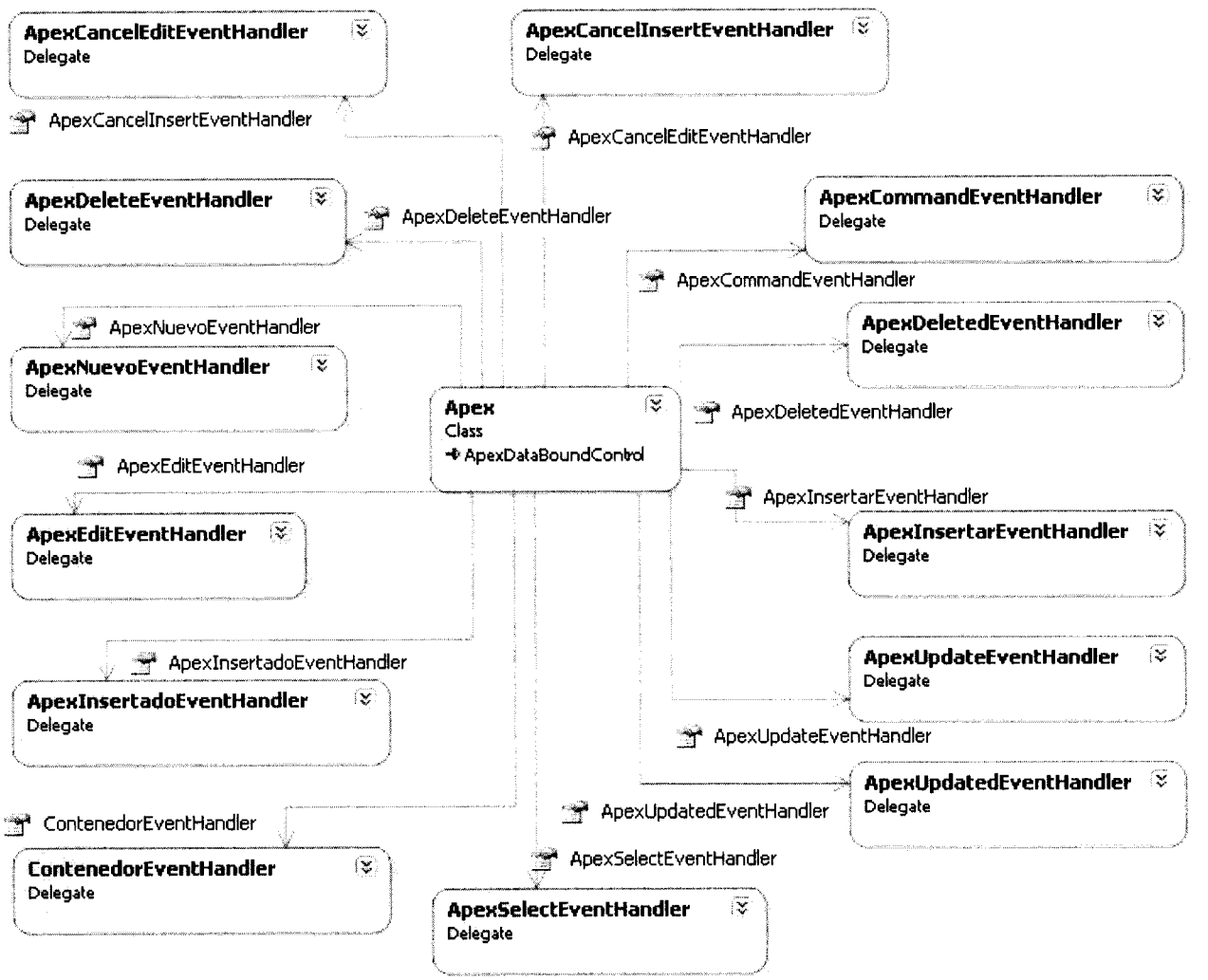
Anexo 3: Diagrama de eventos de la clase Fabrica de la librería JavaScript del control Apex.



Anexo 4: Diagrama de clases del servidor del control Apex.

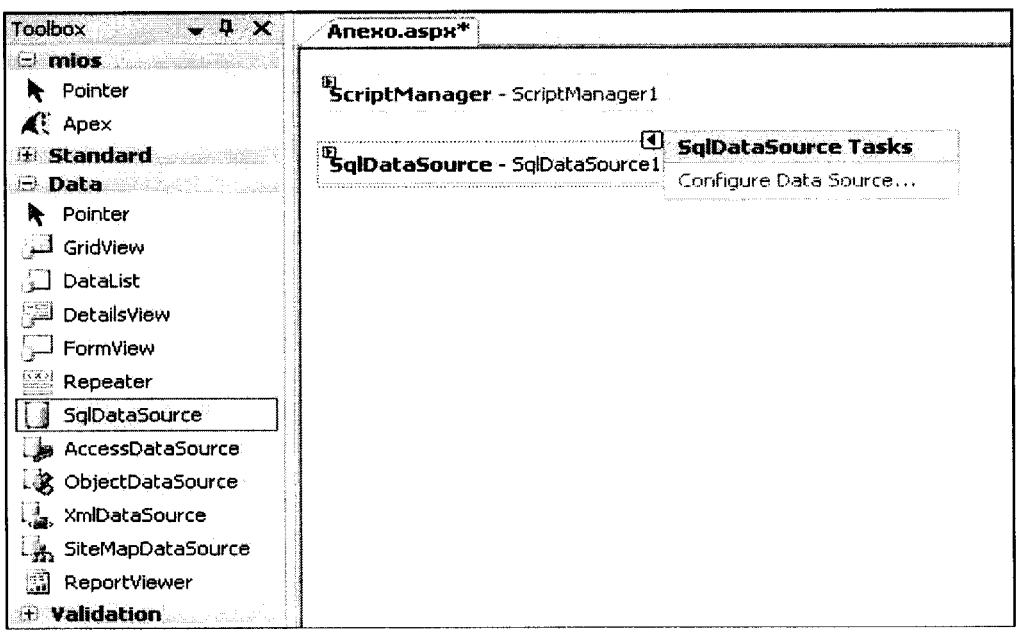


Anexo 5: Diagrama de eventos del servidor para el control Apex.

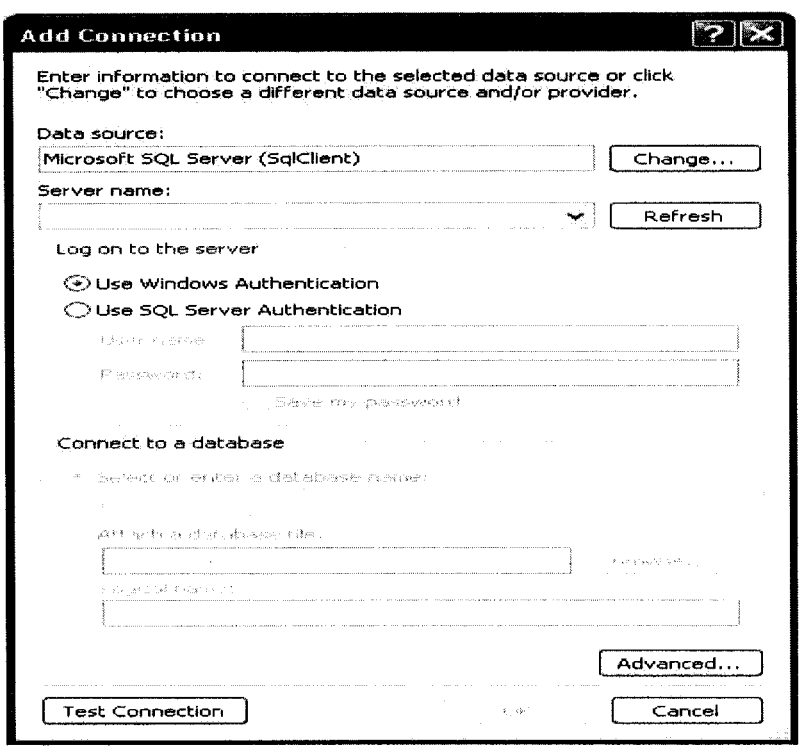


Anexo 6: Configurar fuente de datos.

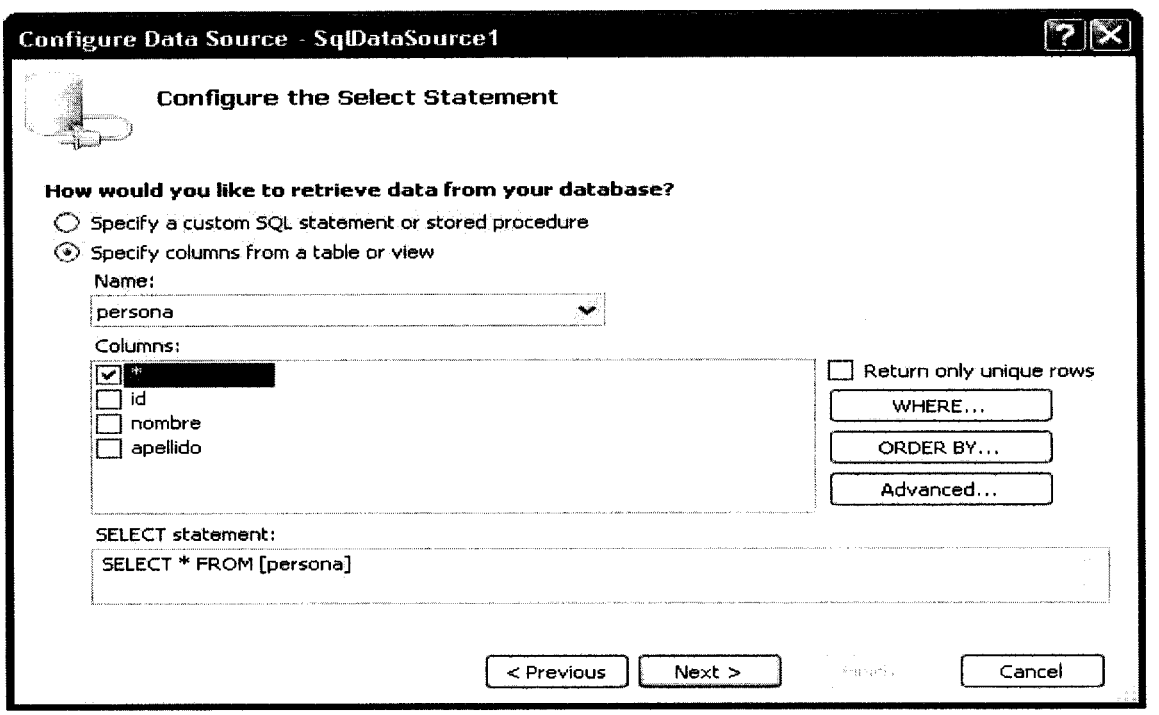
Paso 1: Arrastrar el control de origen de datos a la página web.



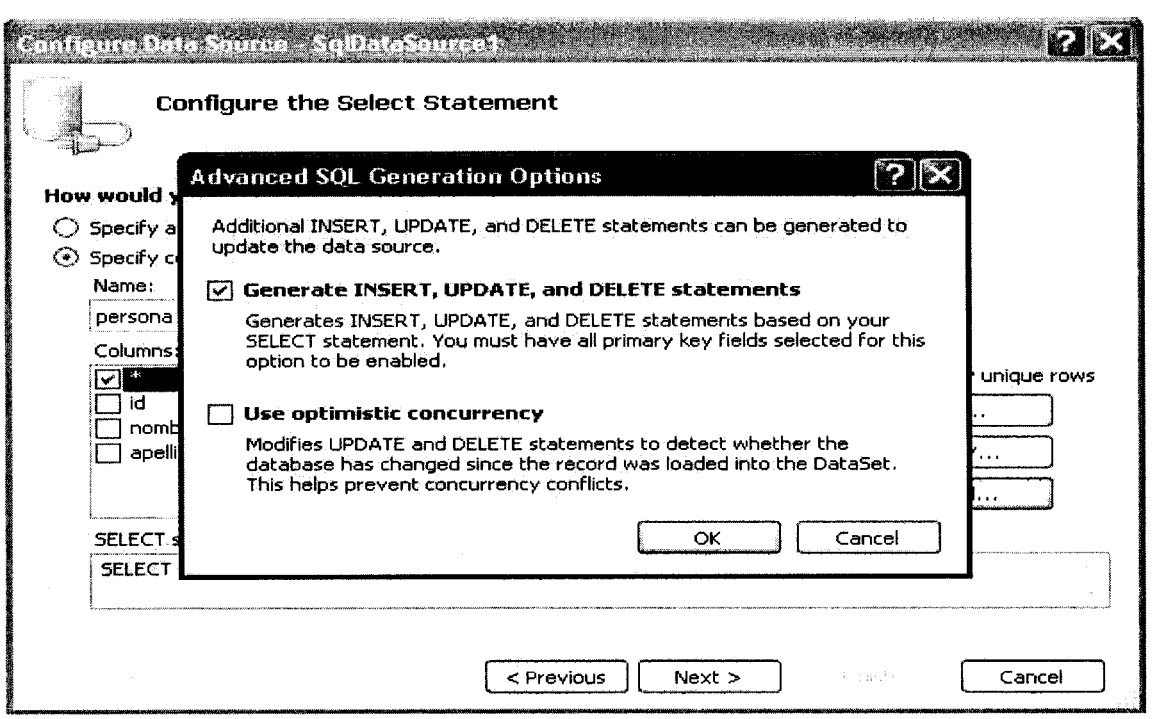
Paso 2: Configurar la conexión a la fuente de datos.



Paso 3: Establecer la consulta para seleccionar los datos.



Paso 4: Permitir eliminar, insertar y actualizar.



Anexo 7: Editar enlace a datos.

Existen dos vías para enlazar un control a datos, la primera es gráficamente a continuación como hacerlo en Apex. Se selecciona el control a enlazar, se despliega el panel de diseño y se selecciona la opción `Edit DataBindings`. **(Figura 1).**

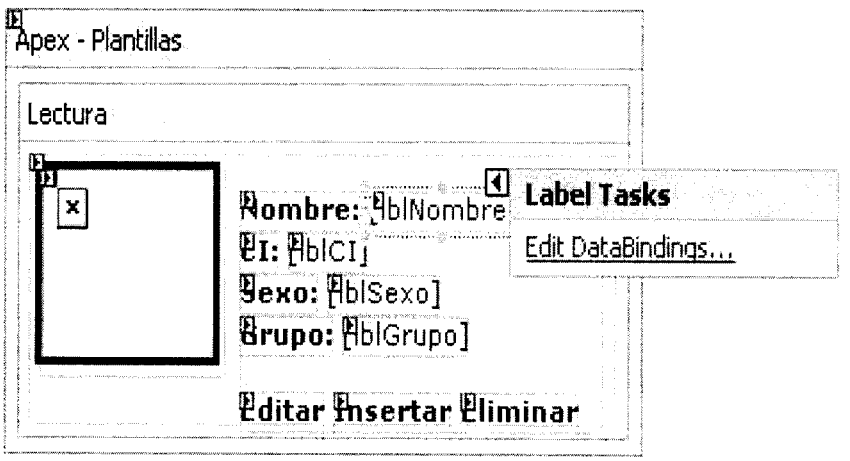


Figura 1: Editar enlace a datos 1.

Seleccionar la propiedad en la parte izquierda del formulario y el campo a enlazar en la lista despegable de la derecha. **(Figura 2).**

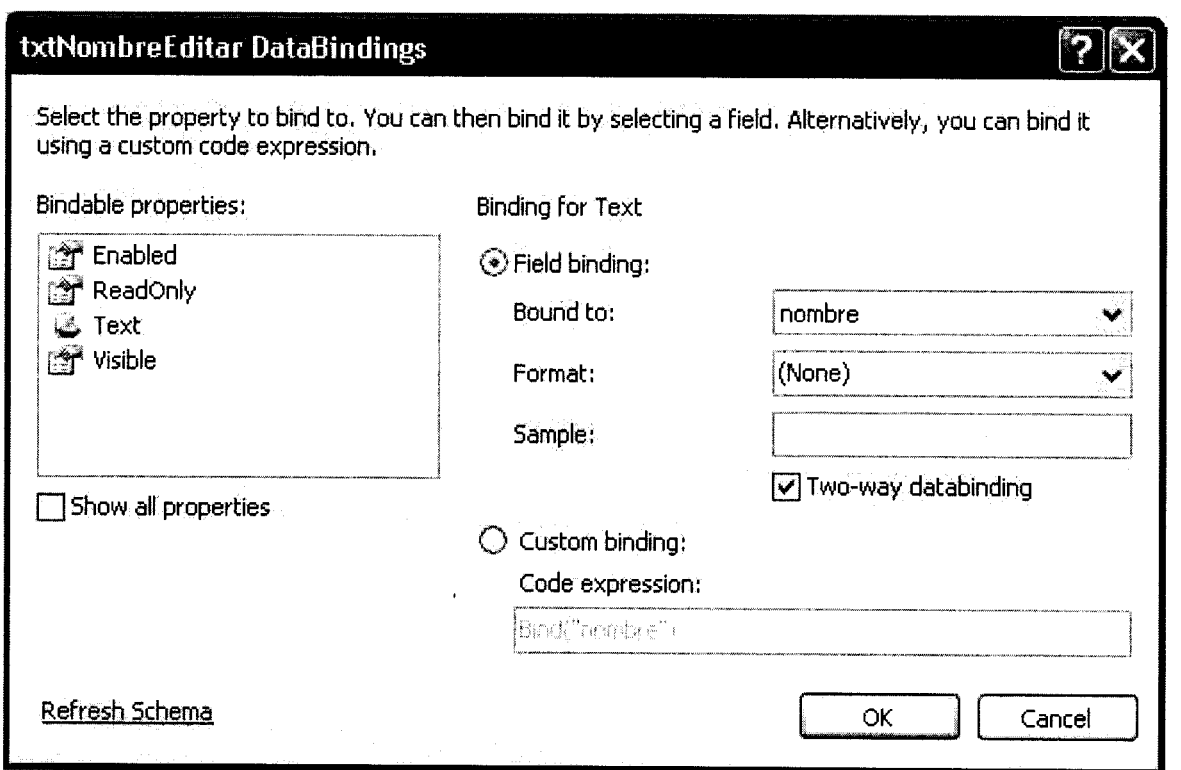


Figura 2: Editar enlace a datos 2.

La segunda manera es definiendo en el código HTML el enlace a datos.

```
<Apex: Apex ID = "Apex1" runat = "server" FuenteDatosVista =
  "SqlDataSource1" Height = 80px          OpacidadHolder = "50" Style
  = "position: relative" TamañoPaginado = "1" Width = "250">
     <LecturaPlantilla>
          <asp: Button ID ="btnAceptar" runat = "server" Style =
"position: relative" Text =                    '<%#Eval ("grupo") %>
'/ >
     </LecturaPlantilla>
</Apex: Apex>
```

Anexo 8: Tabla de propiedades para el ítem.

Propiedad	Descripción	Valores posibles
DesbordeContenedor	Define el comportamiento del contenedor cuando los elementos se desbordan.	Valor de tipo EtipoDesbordeContenedor.
IdContenedor	Identificador del contenedor de los ítem.	ID de un panel o DIV existente en la página.
EstiloItem	Propiedad de tipo ControlStyle que se aplica a cada ítem.	Propiedades de estilo.
MostrarHolder	Define el comportamiento del holder de move.r	True para mostrar siempre. False para mostrarlo en el onmouseover.
TipoHolder	Especifica como se va ha mover el ítem.	Valor de numerador EtipoHolder.
OpacidadHolder	Precisa la opacidad del holder al mover.	Valor numérico de 1 a 100.
UrlImagenHolder	Dirección de la imagen del a mostrar en el holder.	Cadena de dirección.
TipoMovimiento	Define el movimiento del control.	Valor de numerador EtipoMovimiento.

Paginar	Para que el control admita la paginación de los resultados.	True o false.
IndicePagina	Numero de la página a mostrar.	Valor entero mayor que 0.
PosicionPaginado	Indica la posición de la plantilla de paginado.	Valor de tipo ETipoPosicionPaginado.
TamañoPagina	Cantidad de items a mostrar por página.	Valor numérico;
PersistirEstado	Colección de pares propiedad campo a persistir en la base de datos estado.	Objeto de tipo Estado

Tabla: Propiedades del control.

Anexo 9: Enumeradores servidor.

Enumerador	Descripción
ETipoMovimiento	Total : Movimiento sin restricciones. Vertical: El item se mueve verticalmente. Horizontal: El item se mueve horizontalmente. NoMover: El item no se mueve.
ETipoHolder	Icono: El item se mueve por el ícono. Elemento: El item se mueve por el div. Ambos: El item se mueve por el icono y el elemento.
ETipoPaginado	Superior: La plantilla de paginado se muestra como el primer control. Inferior: La plantilla de paginado se muestra como el último control.
EtipoDesbordamiento	Auto: Muestra las barras de desplazamiento si el contenedor es desbordado. Hidden: Si ocurre un desborde del contenedor los ítems desbordados no se muestran. Visible: Si ocurre un desborde se muestra todo el contenido. Scroll: Muestra siempre las barra de desplazamiento.

Tabla: Enumeradores Propiedades.

Anexo 10: Enumeradores cliente.

Enumerador	Descripción
$ETipoMov	Los mismos valores que en el servidor (Anexo 8). Los valores siguientes pueden ser asignados también a las propiedades que reciben este enumerador. total = TTL horizontal = HZL vertical = VTL noMover = NO
$ETipoHolder	Los mismos valores que en el servidor (Anexo 8). Los valores siguientes pueden ser asignados también a las propiedades que reciben este enumerador. icono = 0 elemento =1 ambos = 2

Tabla: Enumeradores cliente.

Anexo 11: Ejemplo de relación.

```
Apex1.Relacion.crear(item1, item2);
Apex1.Relacion.anchoLinea = 1;
Apex1.Relacion.colorLinea  = #000.
```

Para eliminar la relación.

```
Apex1.Relacion.deshacer(item1, item2);
```

Anexo 12: Crear eventos.

Ejemplo de general.

```
function manejador(sender, e)
{      // Implementación. El elemento e, en algunos eventos representa algún
//elemento o una colección de elementos.    }
```

Para el caso de la clase Elemento:

- **Para adicionar el manejador.**

```
NombreControl.Elemento.add[nombre_del_evento]Handler(manejador);
```

- Para eliminar manejador

```
NombreControl.Elemento.remove[nombre_del_evento]Handler(manejador);
```

Para el caso de la clase Fabrica:

- Para adicionar el manejador.

```
NombreControl.Elemento.coleccion["idElemento"].add[
nombre_del_evento]Handler(manejador);
```

- Para eliminar manejador

```
NombreControl.Elemento.coleccion["idElemento"].remove[
nombre_del_evento]Handler(manejador);
```